NBAPI version 2 Guide

For NetBox™ and NetBox Global

March 2023

LenelS2
One Speen Street
Framingham, MA 01701 USA
www.lenels2.com
Technical Support: +1 508.663.2505

Document #API2-UG-1

©2023 Carrier. All Rights Reserved. All trademarks are the property of their respective owners.
LenelS2 is a part of Carrier.

Contents

End-of-Support Notice **1**

Overview **1**
- Database Architecture and the API 2

Enabling NBAPI version 2 in NetBox **2**

Calling the API **2**
- XML Command Calls 3
- XML Responses 4
- Example: Call Command and Response 5
- Buffer Size Allocated for XML Returned by the API 6
- Date Formats 6
- ENCODEDNUM and HOTSTAMP Parameters 6
- Person Access Levels 7
- Time Specs and Time Spec Groups 7
- Errors 8

Setting Up User Roles for the API **10**
- Creating User Roles for API 10

Authentication **11**
- User Login Authentication 11
- Message Authentication 13
- Message Authentication Code 13
- Issuing Partition-Specific API Calls 16

Retrieving and Sending Photo ID Images **17**
- Retrieving JPEG Photo ID Images 17
- Sending JPEG Photo ID Images 18

The Event API (NetBox Only) **20**
- Event Triggering 20
- Streaming Events 20

Intrusion Panel Integration APIs **28**
- Overview of Intrusion Panel Integration Process 28
- Licensing NetBox for NBAPI Intrusion Panels 29
- Understanding NetBox REST API for Intrusion Panel Integration 29
- Adding the LenelS2 NBAPI Panel to NetBox using the Intrusion Panels Configuration Application 30
- Identifier values for Intrusion Panels and associated resources 30
- Creating the Intrusion Panel integration application 30
- REST API Intrusion Panel Integration 37

Using the API with Global **38**
- HTTP Message Address 38
- Authentication 38
- User Roles 38
- Global API Commands 38

Global NBAPI Partition Management 40
Command Reference 42
Actions 42
Configuration 42
Events 42
History 42
People 42
Portals 42
Threat Levels 43
Utility 43
ActivateOutput 44
AddAccessLevel 45
AddAccessLevelGroup 47
AddCredential 49
AddHoliday 51
AddPartition 52
AddPerson 53
AddPortalGroup 58
AddReaderGroup 59
AddThreatLevel 60
AddThreatLevelGroup 61
AddTimeSpec 62
AddTimeSpecGroup 63
AddVirtualCredentialRequest 65
DeactivateOutput 66
DeleteAccessLevel 67
DeleteAccessLevelGroup 68
DeleteHoliday 69
DeletePortalGroup 70
DeleteReaderGroup 71
DeleteTimeSpec 72
DeleteTimeSpecGroup 73
DogOnNextExitPortal 74
GetAccessHistory 75
GetAccessLevel 79
GetAccessLevelGroup 80
GetAccessLevelGroups 82
GetAccessLevelNames 85
GetAccessLevels 87
GetAPIVersion 94
GetCardAccessDetails 95
GetCardFormats 98
GetElevators 99
GetEventHistory 101
GetFloors 103
GetHoliday 105
GetHolidays 107
GetOutputs 108
GetPartitions 110
GetPerson 112

GetPicture 116
GetPortalGroup 118
GetPortalGroups 120
GetPortals 123
GetReader 126
GetReaders 128
GetReaderGroup 130
GetReaderGroups 132
GetTimeSpec 135
GetTimeSpecs 137
GetTimeSpecGroup 140
GetTimeSpecGroups 142
GetUDFLists 144
GetUDFListItems 145
GetVirtualCredentialRequest 147
InsertActivity 148
ListEvents 152
LockPortal 154
Login 156
Logout 158
ModifyAccessLevel 159
ModifyAccessLevelGroup 162
ModifyCredential 164
ModifyHoliday 167
ModifyPerson 169
ModifyPortalGroup 175
ModifyReaderGroup 177
ModifyThreatLevel 178
ModifyThreatLevelGroup 179
ModifyTimeSpec 180
ModifyTimeSpecGroup 182
ModifyUDFListItems 184
MomentaryUnlockPortal 186
PingApp 188
RemoveCredential 189
RemovePerson 191
RemoveThreatLevel 192
RemoveThreatLevelGroup 193
RemoveVirtualCredentialRequest 194
SearchPersonData 195
SetThreatLevel 202
StreamEvents 203
SwitchPartition 205
TriggerEvent 206
UnlockPortal 208
Deprecated Commands 209

End-of-Support Notice

Effective November 2022, beginning with the release of NetBox™ 5.6, NetBox API (NBAPI) version 1 will be deprecated (no longer receive support or updates) and replaced by NBAPI version 2. NBAPI version 1 will be retired in NetBox 5.8 (approximately November 2023).

Overview

This guide describes the NetBox NBAPI version 2, which permits network-connected systems to perform various operations with a NetBox under program control. This guide also includes a limited subset of APIs for NetBox Global™. This document refers to NetBox and Global collectively as "the system," except when it is necessary to distinguish between the two products.

The information in this document is current with NetBox Release 5.4.x and Global 2.19 or above.

NBAPI version 2 allows NetBox and Global systems to integrate with third-party systems and other proprietary software applications such as HR/payroll, directory services, visitor management, building management, and event monitoring.

All components managed by the system are network-connected and managed through a web-based user interface.

NBAPI version 2 can perform the following operations:

- Add, modify, delete, and retrieve information on:
  - Person records and search-directed person record information.
  - Credentials, portals, time specs, readers, threat levels, holidays and their groups.
  - Access levels.
- Add a partition.
- Switch between partitions and perform operations on the specified partitions.
- List and trigger events.
- Provide notification when events occur.
- Activate, deactivate, and retrieve information on outputs.
- Upload and download photo ID images.
- Lock and unlock portals.

Important: If NBAPI version 2 is used to update person attributes and access level assignments that are managed on a Microsoft Active Directory (AD) server, the updates will be overwritten the next time this data is synchronized—either when the data is changed on the AD server or when the next full synchronization is performed. For this reason, we do not recommend using NBAPI version 2 to update data on a NetBox that synchronizes its data with an AD server.

A full synchronization is performed whenever communication with the AD server is stopped and restarted, and whenever the AD server configuration is disabled and re-enabled.

Database Architecture and the API

With the API, users can retrieve information from the system database and issue commands that modify database tables.

The system database includes a table of person records that includes attributes such as access levels, which specify access privilege, and to access cards, which are the credentials used for access control.

The system database also includes tables of attributes that control the administrative, monitor, and maintenance features of a NetBox or Global system.Enabling NBAPI version 2 in NetBox

Enabling NBAPI version 2 in NetBox

To enable NBAPI version 2 in NetBox, perform these steps:

1. In the NetBox GUI, select *Configuration : Site Settings : Network Controller* and select the **Data Integration** tab.
2. Select the **Enable V2** check box.
3. Click **Save**.

Calling the API

The API is invoked by posting an HTTP request to the web server on the system. The request consists of the XML that describes the API command.

API requests are at the following addresses:

- All NetBox API commands, except those for the Global API:

  http://<NetBox IP address>/nbws/goforms/nbapi

- Event NetBox API commands for retrieving the event list, triggering an event, and receiving the event stream:

  http://<NetBox IP address>/nbws/goforms/nbapi

- Global API commands:

  http://<Global IP address>/global/goforms/nbapi

The IP port for these commands is the port at which the web server responds. The default is port 80. If you are using a different web port, such as 8080, you must specify it as part of the IP address (for example, 102.168.0.220:8080).

XML Command Calls

The XML API command passed to the server is wrapped in a `NETBOX-API` element block. Within `NETBOX-API` element block is a `COMMAND` element block. The `COMMAND` element block includes a command name attribute (name="*command-name*") and a block of parameters enclosed within a `PARAMS` element block.

IMPORTANT: All XML tags must be in uppercase.

You can use authentication based on login with username and password using the `SESSIONID` element or MAC authentication using the `MAC` element.

The structure of the XML command using a session ID is:

```
<NETBOX-API sessionid="session ID returned in Login command">
     <COMMAND name="command name" num="1" dateformat="tzoffset">
        <PARAMS>
              <element>element value</element>
                                 .
                                 .
                                 .
              <element>element value</element>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

The structure of the XML command using MAC authentication is:

```
<NETBOX-API>
     <COMMAND name="command name" num="1" dateformat="tzoffset">
        <PARAMS>
              <element>element value</element>
                                 .
                                 .
                                 .
              <element>element value</element>
        </PARAMS>
     </COMMAND>
     <MAC>Message Authentication Code</MAC>
</NETBOX-API>
```

The command includes a number of attributes. The attribute values must be surrounded by quotes (" "):

- `SESSIONID` (use for username and password authentication): Alphanumeric value is returned by the **Login** command. The `SESSIONID` must be included in each subsequent command call, including the **Logout** command.

  An API session will remain active until a **Logout** command is issued or until the session times out after the *Session Timeout* threshold configured on the system is reached. For NetBox, the *Session Timeout* value is set on the *Web Site* tab of the *Network Controller* page. For Global the timeout is set to 5 minutes.

  For NetBox, Username and password authentication requires that the *Use Authentication* and the *Use login username/password for authentication* check boxes are selected on the *Data Integration* tab of the *Network Controller* page. For Global login authentication is always required.

- `COMMAND` name: `Command name.`

- `num`: Identifier for the `COMMAND` element block.

  The `num` attribute is always set to 1 because multiple `COMMAND` element blocks in a single XML message are not supported.

- `dateformat` (optional): If `dateformat` is supplied, dates are returned with an additional timezone offset ("`tzoffset`") which returns the value in GMT format.

  See *Date Formats* for details.

- `MAC` (use only for MAC authentication for NetBox only): `MAC` element entry containing the computed message authentication code for the entire contents of the XML message.

  MAC authentication requires that the *Authentication* check box is selected and the *Use login username/password for authentication* check boxes is **not** selected on the *Data Integration* tab of the *Network Controller* page.

  For additional information on MAC authentication, see *Message Authentication Code.*

Note: XML tags that are not supported by the API but are syntactically valid XML are ignored.

XML Responses

The XML response contains the following elements and attributes:

- `NETBOX`: the outermost element of the response
  - "sessionid=" attribute identifying the session associated with the logged in user.
- `RESPONSE`: element block containing results of command which may include:
  - `<APIERROR>` indicates a failure occurred at the API level. No additional elements or attributes are contained in the response. See APIERROR Values for an explanation of the failure.
  - "command=" attribute indicating the type of command the response applies to.
  - "num=" attribute indicating the instance of the command.
  - `<CODE>` element with value of either `SUCCESS`, `FAIL`, or `NOT FOUND`:
    - `SUCCESS`: Command successfully completed and may return additional elements.
    - `FAIL`: Command failed and may return an error message in the `ERRMSG` element contained in the `DETAILS` element block.
    - `NOT FOUND`: Indicates that some element of the API request count not be found. For example, the specified element does not exist.

      `<DETAILS>` element block that includes additional details regarding the command response.

The structure of a successful XML response using `SESSIONID`:

```
<NETBOX sessionid="session ID returned in Login command">>
    <RESPONSE name="command-name" num="1">
       <CODE>SUCCESS </CODE>
       <DETAILS>
          <element block>
             <element>value</element>
```

```
                                    .
                                    .
                                    .
                <element>value</element>
            </element  block>
            <element  block>
                <element>value</element>
                                    .
                                    .
                                    .
                <element>value</element>
            </element block>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

Example: Call Command and Response

This is an example of a **GetPerson** call which requests information for a person record with a person ID of 21001.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetPerson" num="1" dateformat="tzoffset">
        <PARAMS>
            <PERSONID>21001</PERSONID>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

This successful response to the **GetPerson** command response returns the person record definition.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPerson" num="1">
        <CODE>SUCCESS</CODE>
            <DETAILS>
            <PERSONID>21001</PERSONID>
            <FIRSTNAME>Isaac</FIRSTNAME>
            <MIDDLENAME>P</MIDDLENAME>
            <LASTNAME>Johnson</LASTNAME>
            <EXPDATE>2024-01-01 00:00:00</EXPDATE>
            <ACTDATE>2016-09-11 00:00:00</ACTDATE>
            <NOTES>Isaac drives a motorcycle</NOTES>
            <UDF1>Added User Defined Field 1</UDF1>
            <UDF2>Added User Defined Field 2</UDF2>
            <UDF3>Added User Defined Field 3</UDF3>
            <UDF4>Added User Defined Field 4</UDF4>
            <UDF5>Added User Defined Field 5</UDF5>
            <UDF6>Added User Defined Field 6</UDF6>
            <UDF7>Added User Defined Field 7</UDF7>
            <UDF8>Added User Defined Field 8</UDF8>
            <UDF9>Added User Defined Field 9</UDF9>
            <UDF10>Added User Defined Field 10</UDF10>
            <PICTUREURL>isaac_johnson.jpg</PICTUREURL>
            <DELETED>FALSE</DELETED>
            <ACCESSLEVELS>
                <ACCESSLEVEL>My NetBox access level 1</ACCESSLEVEL>
                <ACCESSLEVEL>My NetBox access level 2</ACCESSLEVEL>
            </ACCESSLEVELS>
        </DETAILS>
```

```
    </RESPONSE>
</NETBOX>
```

The following example shows an unsuccessful response resulting from an authentication failure (such as failing to check the *API Enabled* checkbox on the *Network Controller* page):

```
<NETBOX>
  <RESPONSE>
    <APIERROR>5</APIERROR>
  </RESPONSE>
</NETBOX>
```

Note: See *APIERROR Values* for the list of `APIERROR` values and descriptions.

Buffer Size Allocated for XML Returned by the API

The client programs that call the API may need to increase the maximum buffer size they allocate for the XML message returned by the API to function correctly.

The maximum size of a picture that can be returned by the API is 120KB in GetPerson.

Date Formats

There are different rules for supplying dates as input, and the dates that are retrieved. All dates are in the local time of the system.

- Activation date and expiration dates: Activation date (ACTDATE) and expiration date (EXPDATE), if provided in the **AddPerson** and **ModifyPerson** commands, must be in one of the following forms:

  ```
  YYYY-MM-DD HH:MM
  YYYY-MM-DD
  ```

  If the time is not provided, the default is 00:00.

- Returned dates are shown in two forms:

  If the command does not have the '`DATEFORMAT="TZOFFSET"`' attribute, the dates are returned by default as:

  ```
  YYYY-MM-DD HH:MM:SS
  ```

  If '`DATEFORMAT="TZOFFSET"`' is supplied as an attribute to the command, then the returned value is in GMT format:

  ```
  YYYY-MM-DD HH:MM:SS +/-HHMM
  ```

  For example, 5am EST would be returned as:

  ```
  2016-10-31 10:00:00 -0500
  ```

ENCODEDNUM and HOTSTAMP Parameters

ENDCODEDNUM is a representation of the actual data on the credential. Credentials are interpreted using a set of rules defined by the card format type. The ENCODEDNUM value passed to the API is a string of ASCII characters as decimal digits '0'-'9'. Some card formats may also support hex digits 'A'-'F'. The string value passed depends on the card format type and adheres to a maximum length. A list of the card format names can be retrieved in the API using the GetCardFormats command.

Some commands provide the option of supplying an ENCODEDNUM value and a HOTSTAMP value.

HOTSTAMP is a value optionally stamped on the card or recorded for reference. Some deployments will choose to have these fields use the same value. In the API, if only ENCODEDNUM or HOTSTAMP is supplied, that value will be used for both the ENCODEDNUM and HOTSTAMP.

Person Access Levels

The syntax below describes two forms of syntax for adding and removing access levels to/from person records. Both forms can be used in the AddPerson and ModifyPerson commands.

The first form, Access Levels by Name, specifies one or more access level names to be added to a person. If one or more access levels are passed with the ModifyPerson command, the API will first clear all existing access levels from the Person record before adding new ones. In other words, this is a replace operation. Thus, all access levels that should continue to be associated with a Person record must be specified in a call to ModifyPerson.

The second form, Access Levels by Name and Flag, adds or deletes one or more access levels from a person. This form is referred to as "Alternative Syntax."

Access Levels by Name only:

```
<ACCESSLEVELS>
    <ACCESSLEVEL>access level 1</ACCESSLEVEL>
    <ACCESSLEVEL>access level 2</ACCESSLEVEL>
          .
          .
          .
    <ACCESSLEVEL>access level 32</ACCESSLEVEL>
</ACCESSLEVELS>
```

Access Levels by Name and Flag (Alternative Syntax):

```
<ACCESSLEVELS>
    <ACCESSLEVEL>
       <ACCESSLEVELNAME>UI Access Level 2</ACCESSLEVELNAME>
       <DELETE>1</DELETE>
    </ACCESSLEVEL>
    <ACCESSLEVEL>
       <ACCESSLEVELNAME>UI Access Level 3</ACCESSLEVELNAME>
       <DELETE>0</DELETE>
    </ACCESSLEVEL>
</ACCESSLEVELS
```

A value of '1' for `<DELETE>` will result in an access level being removed from the Person record. A value of '0' will result in an access level being added to the Person record. The `DELETE` tag with a value of 1 can be used only with the ModifyPerson command. Any access level already associated with a Person record but not specified in a call to ModifyPerson will continue to be associated with that Person record.

Time Specs and Time Spec Groups

A time spec group is constructed out of one or more time specs. The system is initialized with two time specs and two corresponding time spec groups: **Always** and **Never**.

When a time spec is modified or deleted, any time spec group of which that time spec is a member is also modified or deleted.

Although you can construct a new time spec group out of any combination of existing time specs, you cannot modify or delete any of the singular time spec groups that either existed at system initialization time.

Time spec groups are returned in a call to the **GetTimeSpecGroups** command.

Errors

APIERROR Values

A failure at the API level will have one of the following values in the APIERROR element.

APIERROR Code	Description
1	The API failed to initialize.
2	The API is not enabled on the system. To enable API on your NetBox Configuration, select the *Enabled* check box on the *Data Integration* tab on the *Network* Controller page.
3	The call contains an invalid API command.
4	The API was unable to parse the command request.
5	There was an authentication failure. Refer to options for configuring authentication to work with the API.
6	The XML code contains an unknown command. Check the syntax of the command request.

The following example shows a **PingApp** call which results in a response which includes an `APIERROR` number.

```
<NETBOX-API>
     <COMMAND name="PingApp" num="1">
     </COMMAND>
</NETBOX-API>
```

The response contains APIERROR 5 indicating that there is an authentication failure.

```
<NETBOX>
     <RESPONSE>
        <APIERROR>5</APIERROR>
     </RESPONSE>
</NETBOX>
```

Error Messages

A failure at the command level will result on one of the following elements in the CODE element:

- FAIL: Returns DETAILS which may include one of the following elements:
  - ERRMSG: A text error message. See ERRMSG Description Values.
  - NOT FOUND: Indicates that the API could not find a response.

Example

The following example shows an **AddCredential** call that results in a response that includes an ERRMSG, because there was no person record with ID number: 6342.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
   <COMMAND name="AddCredential" num="1">
       <PARAMS>
            <PERSONID>6342</PERSONID>
            <CARDFORMAT>26 bit Wiegand</CARDFORMAT>
            <HOTSTAMP>1111</HOTSTAMP>
            <ENCODEDNUM>1111</ENCODEDNUM>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The response includes the ERRMSG, "Person not found".

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="AddCredential" num="1">
       <CODE>FAIL</CODE>
       <DETAILS>
          <ERRMSG>Person not found</ERRMSG>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

The response includes the ERRMSG, "Invalid PersonID".

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="AddCredential" num="1">
       <CODE>FAIL</CODE>
       <DETAILS>
          <ERRMSG>Invalid PersonID</ERRMSG>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

Type Element Values

A TYPE element identifies an activity type and is returned in the DETAILS element. The TYPE element is used in commands such as **GetAccessHistory** and **GetCardAccessDetails**.

Type Value	Description
1	Valid access (no reason value in response)
2	Invalid access
37	Elevator valid access (no reason code value in response)
38	Elevator invalid access
64	Access not completed

Reason Element Values

A `REASON` element provides additional information about an invalid activity type and is returned in conjunction with an element `TYPE` in the `DETAILS` element. The `REASON` element is used in commands such as **GetAccessHistory** and **GetCardAccessDetails**.

Reason Value	Description
1	Card not in local database
2	Card not in S2NC database
3	Wrong time
4	Wrong location
5	Card misread
6	Tailgate violation
7	Anti-passback violation
8	--unused--
9	Wrong day
10 – 13	--unused--
14	Card expired
15	Card bit length mismatch
16	Wrong Day
17	Threat Level (prevented access)

Setting Up User Roles for the API

A user role defines a specific set of permissions that can be assigned to a user to allow him or her access to resources on a system. A NetBox system comes preconfigured with the following user roles:

- *Full system setup*: This role has full read/write API access for all partitions (in a partitioned system).
- *Partition setup*: This role has read/write API access for the partition.
- *Partition administrator*: This role does not have API access.
- *Partition monitor*: This role does not have API access.

Creating User Roles for API

Custom user roles can be created and assigned to users with the setup privilege that allows them to run API programs on their native partitions. This access is based on the type of commands used in the API program:

- Read-Access commands retrieve information from a NetBox system, such as **GetReader**, **SearchPersonData** and **GetAccessHistory**.

- Write-Access commands, such as **ModifyPerson**, **ActivateOutput, and LockPortal** perform actions that make changes to a system, such as modifying person records, activating outputs, or locking and unlocking portals.

To create a custom user role that enables users to run API programs on a single partition, based on command type:

1. Select *Configuration : Site Settings : User Roles*.
2. Under API Privilege, select one of the following Access options:
   - *Read-Only*: Users with this role will be able to run API programs with Read-Access commands.
   - *Read-Write*: Users with this role will be able to run API programs with Read-Access and Write-Access commands.
3. (optional) Under Security, select *Restrict User to API Login only* if you want users with this role to be able to log into the NetBox via the API, but not via the web interface.
4. Select ***Save***.

For example, suppose you have a program that provides monitoring functions using Read-Access commands. You could create a user role for the application and specify *Read-Only* in the *API Privilege* section. Assign the role to a person and use that person's login credentials for the application.

You could create another user role for an API program that issues Read-Write commands to activate outputs and lock/unlock portals. For this program, you could create a user role and specify *Read-Write* in the *API Privilege* section.

Authentication

There are two modes of authentication available for accessing the API:

- User login authentication
- MAC authentication

There are some functions that require user login authentication.

User Login Authentication

User login authentication requires a username and password for authentication for programs using the API.

To set up user login authentication for the API on NetBox:

1. Select *Configuration : Site Settings : Network Controller*.
2. Select the *Data Integration* tab.
3. Select the *Enabled* check box,
4. Select the *Use Authentication* check box.
5. Select the *Use login username/password for authentication* check box.

Note: As of Release 5.4 and above, the login response includes a cryptographic hash value rather than an integer for the NetBox session ID—for example:

```
<NETBOX
sessionid="9wK4I20DsnhJhL9srYco36P1vaOr4EDNgFJAwxHwwWtAi8WhJ7W7p2u42brOdAuS">
```

This value must be carried through to all subsequent requests. Note that some users (such as those working in a language that does not permit an alphanumeric hash to be stored in an integer) might need to allocate a different and longer value to store the session ID.

To log in using the API, you need to include a **Login** command in your program code. A **Login** call must always be matched by a corresponding **Logout** call before the program terminates.

When user login authentication is enabled, the API session will timeout when the *Session Timeout* value is reached in the NetBox user interface. The *Session Timeout* value is set on the *Web Site* tab on the *Network Controller* page.

When a session times out, a **Login** call must be submitted to open a new API session.

Further, the parameter *Limit Session to Single IP Address*, which defaults to selected, will ensure that if an IP address changes during a session that the user will be required to login again. Changing this to *NO* will allow more than one IP address to participate in a single session, which is less secure.

Here is an example of a **Login** call:

```
<NETBOX-API>
  <COMMAND name="Login" num="1" dateformat="tzoffset">
    <PARAMS>
      <USERNAME>admin</USERNAME>
      <PASSWORD>admin</PASSWORD>
    </PARAMS>
  </COMMAND>
</NETBOX-API>
```

A successful response to the **Login** command has the following form:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
  <RESPONSE command="Login" num="1">
    <CODE>SUCCESS</CODE>
  </RESPONSE>
</NETBOX>
```

The `sessionid` returned by the call to the **Login** command must be included within the `NETBOX-API` element tag for all subsequent calls to the API.

Here is an example of an **AddPerson** command using the `sessionid` from the **Login** command response:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="AddPerson" num="1" dateformat="tzoffset">
    <PARAMS>
      <LASTNAME>Jet</LASTNAME>
      <FIRSTNAME>Joan</FIRSTNAME>
    </PARAMS>
  </COMMAND>
</NETBOX-API>
```

Here is an example of a successful response to the **AddPerson** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
  <RESPONSE command="AddPerson" num="1">
    <CODE>SUCCESS</CODE>
    <DETAILS>
```

```
      <PERSONID>_6</PERSONID>
    </DETAILS>
  </RESPONSE>
</NETBOX>
```

The session is terminated with a call to **Logout** with the `sessionid` for the session being terminated**.**

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="Logout" num="1" dateformat="tzoffset"/>
</NETBOX-API>
```

Here is an example of a successful response to a **Logout** call.

```
<NETBOX>
  <RESPONSE command="Logout" num="1">
    <CODE>SUCCESS</CODE>
  </RESPONSE>
</NETBOX>
```

Message Authentication

Each message sent by the caller contains a sequence number that is part of the message authentication code in the MAC element. It is up to the caller to increment the sequence numbers. The NetBox remembers the highest sequence number that is has seen, and will reject messages that have sequence numbers less than or equal to that number.

The system administrator has the option of turning off message authentication from the NetBox user interface if there are other methods of data security in place.

It is up to the caller to remember the highest sequence number transacted, and to always transact sequence numbers higher than the last one. If that becomes impossible, you can select the *Reset Sequence Number to "0"* check box in the NetBox user interface to reset the sequence numbering to zero (0). This check box is located on the *Data Integration* tab of the *Network Controller* page.

Message Authentication Code

Message Authentication Code (MAC) uses the SHA (Secure Hash Algorithm), and message sequence numbers to ensure the security and integrity of the system.

Message Authentication Code is based on the SHA-1 hashing algorithm that is designed to be unique for every message and impossible to reverse in practice.

The message authentication code is transmitted in the `MAC` element of the message and has the form:

RAN1 (1-5)	RAN2 (6-10)	SEQ # (11-20)	SHA-1 result as 40 hexadecimal digits (21-60)

Where:

- RAN1 and RAN2: Two random numbers (each one is five decimal digits).
- SEQ #: Zero-padded 10-digit sequence number that is incremented for each message transmitted by the caller.
- SHA-1 hash result: Remaining message which consists of a 40-hex digit SHA-1 hash as produced by the algorithm provided.

Each API command is accompanied by a message authentication code, which includes a sequence number. The message authentication code must be correct and the sequence number must be greater than sequence number of the previous command or the system will ignore the message.

Generating the MAC

Code in C is provided by LenelS2 and other organizations to calculate the SHA-1 digest used in the MAC.

LenelS2 also provides a utility function, `generate_mac`, to create the MAC from caller-provided inputs. You can find the `generate_mac` utility function by going to *www.s2sys.com*, signing into Support Central, and clicking the link for the *Software Developers* page.

The following steps describe how to use the *generate_mac* function.

Step 1: To generate a MAC for the message to send to the API, your code needs to supply the following:

- `msg`: XML message you wish to send. `msg` must be null terminated and should not contain the `<MAC>...</MAC>` tags.
- `seq`: Sequence number of the message: it should be GREATER than the sequence number stored on the system, which is the sequence number of the previous message.
- `secret`: A null terminated string with exactly 8 characters with no trailing blanks, the same as that stored on the NetBox.
- `rand1`: Random number generated (e.g., rand1 = random()
- `rand2`: Random number generated (e.g., rand2 = random()

Step 2: With these input parameters, call the `generate_mac` function, which has the following signature:

```
unsigned char *generate_mac(unsigned char *msg,
                            unsigned long seq,
                            unsigned char *secret,
                            unsigned short rand1,
                            unsigned short rand2,
                            unsigned char *mac)
```

Note: The `char *mac` returned by the `generate_mac` function is the 60-character Message Authentication Code that you insert into the XML message you wish to send to the system.

The `generate_mac` does the following:

1. Concatenates `rand1`, `rand2` (each value converted to 5-decimal digits), and `seq` (represented as 10-decimal digits) into a 20-decimal digit seed buffer called seed.
2. Generates a check-sum of the `msg` called `chksum`.
3. Creates `buffer`, which is a concatenation of `chksum`, `secret`, and `seed`.
4. Passes `buffer` to a function that implements the SHA-1 hashing algorithm, which returns a 40-hex digit SHA-1 hash called `sha`.
5. Concatenates `seed` and `sha` to produce `mac`, which is returned as a 60-character string.

Your code will then:

1. Add the XML tags with the `mac` value to create mac1 (<MAC>mac</MAC>).
2. Remove the `NETBOX-API` tags from `msg` to create `msg1`.

3. Concatenate `msg1` with `mac1` to create `msg2`.
4. Add the `NETBOX-API` tags to `msg2` to create `msg3`.
5. Send `msg3` to the API.

When the API on the system receives this message, it does the following:

1. Extracts the random numbers, the sequence number, and `sha` from `mac`.
2. Repeats the calculation on the incoming message (minus `mac1`) using the secret stored on the system using the random numbers and sequence number.
3. Compares the resulting `SHA` with the `SHA` from the `MAC`. If they are the same, the authentication succeeds.

Sequence Numbers

The system maintains its own sequence number, which is initially set to zero by default. It may be reset to zero at any time via the NetBox web interface (*Configuration : Site Settings : Network Controller*).

The system compares the sequence number of every incoming message with its own copy. If the former is greater, the system updates its copy with the new value, and the message continues to be processed. Otherwise, the message is rejected.

Example

Generated MAC in an **AddPerson** call when authentication by MAC is enabled:

```
<NETBOX-API>
  <COMMAND name="AddPerson" num="1" dateformat="tzoffset">
    <PARAMS>
      <LASTNAME>Jet</LASTNAME>
      <FIRSTNAME>Joan</FIRSTNAME>
      <ACCESSLEVELS>
        <ACCESSLEVEL>Access Level1</ACCESSLEVEL>
      </ACCESSLEVELS>
    </PARAMS>
  </COMMAND>
  <MAC>33464016690000000000131B2774C86CDB2CAEE66A72B9E2521451473CE85</MAC>
</NETBOX-API>
```

Successful response to the **AddPerson** call:

```
<NETBOX>
  <RESPONSE command="AddPerson" num="1">
    <CODE>SUCCESS</CODE>
    <DETAILS>
      <PERSONID>_6</PERSONID>
    </DETAILS>
  </RESPONSE>
</NETBOX>
```

An unsuccessful response resulting from an authentication failure (such as failing to check the *API Enabled* checkbox on the *Network Controller* page):

```
<NETBOX>
  <RESPONSE>
    <APIERROR>5</APIERROR>
  </RESPONSE>
</NETBOX>
```

Issuing Partition-Specific API Calls

To make API calls that affect a partition, *Use login username/password for authentication* must be enabled on the *Data Integration* tab of the *Network Controller* page. In addition, the appropriate user role must be assigned to the user.

- To access a specific partition, the user must have the *full system setup user role*, *the partition setup user role in that partition*, or a custom user role with *API Privilege*s in that partition.
- To switch to a non-native partition, a user with either the partition setup user role or custom user role with API privileges in his native partition must have this user role mapped to a user role in a non-native partition that allows that same access. Subsequent commands will apply to the non-native partition. See online help for details about role mapping (search for *Granting User Roles to Users in Other Partitions*).
- There are two methods of making API calls to affect a partition:
    - Issue a Login call to the user's native partition. All subsequent API calls will apply only to the native partition.
    - Issue a **Login** call to a user's partition and a **SwitchPartition** call to switch the partition. All subsequent API calls will apply *only* to the switched partition.

For example, if you want to set a threat level in the Master partition or a user's native partition:

1. Issue a **Login** call using the username and password of a user with *Full Setup* role, *Partition Setup* role, or a custom user role with API command Read-Write privilege in the native partition.
2. Issue a **SetThreatLevel** call with the login response session ID.
3. Issue a **Logout** call with the session ID.

If you want to set a threat level in an alternate partition:

1. Issue a **Login** call using the username and password of a user with *Full Setup* role, *Partition Setup* role, or a custom user role with API command Read-Write privilege in the native partition.
2. Use the session ID returned in the **Login** call in all subsequent API calls.
3. Issue a **GetPartitions** call to retrieve the list of partition keys.
4. Issue a **SwitchPartition** call and include the partition key.
5. Issue a **SetThreatLevel** call.
6. Optionally, switch back to the Master or native Partition.
7. Issue a **Logout** call.

Retrieving and Sending Photo ID Images

There are two methods for working with photo ID images (pictures) associated with a person:

- API Commands: **AddPerson**, **ModifyPerson** and **GetPicture.**

  When using these commands, data is passed as a Base64 encoded string.

  - The **AddPerson** and **ModifyPerson** commands are used to upload (send) a person's photo ID image to the NetBox.
  - The **GetPicture** command is used to download (retrieve) a person's photo ID image from the NetBox.

    The image size is limited to a maximum of 120KB when using these commands.

- Direct URL:
  - Use **HTTP POST** to send (upload) a photo ID image to a NetBox system.
  - Use **HTTP GET** to retrieve (download) a photo ID image from a NetBox system.

    Image sizes up to a maximum of 1024KB can be sent and retrieved from a NetBox system using **HTTP POST** and **HTTP GET** requests. The maximum size allowed by these requests depends on the value configured in the *Photo ID Size Limit* drop-down menu from the *Admin* tab on the *Network Controller* page.

Retrieving JPEG Photo ID Images

The **GetPicture** command returns a Base64 encoded string representing a person's picture.

The following procedure uses the Direct URL method.

To retrieve photo ID images that were previously uploaded to a system:

1. Use the **Login** command to retrieve a session identifier.

   Use the **SearchPersonData** command to retrieve a list of person records. Inside the response for each person record is a `PICTUREURL` element. The content of this element holds the file name of the photo identification picture of the person.

2. Use an **HTTP GET** request specifying the following url:

   `http://<NetBox host IP address>/upload/pics/<person record pictureurl filename>`

   The **HTTP GET** request must specify a cookie in the HTTP header with the following contents:

   `.sessionid=<session identifier string retrieved from Step 1>`

   Note: A dot (.) is required before `sessionid`.

   The **HTTP GET** request will download the specified photo ID image.

The following program code provides a sample procedure that retrieves photo ID images.

```
public void DownloadPicture(string sPictureFileName)
 {
    // ex: http://10.160.20.21/upload/pics/mystyle.jpg
    try
    {
       WebClient = new WebClient ();
       byte [] imageBytes = null;
       Uri = new Uri ("http://" + mHostAddress + "/upload/pics/" +
                         sPictureFileName);
       string simpleCookie = ".sessionId="+mSessionId;
       webClient.Headers.Add("Cookie", simpleCookie);
       imageBytes = webClient.DownloadData(uri);
       if (imageBytes.Length > 0)
       {
            System.IO.File.WriteAllBytes(sPictureFileName, imageBytes);
       }
    }
    catch (Exception exp)
    {
       string sMessage = exp.Message;
    }
 }
```

Sending JPEG Photo ID Images

The following code sample demonstrates how to send (upload) upload a person's photograph using a valid login session and a multi-part HTTP file upload request to the NetBox.

> **Note**: The buffer parameter holds the byte contents of a JPEG image (the only supported image format).

```
private bool UploadPhoto (byte[] buffer, string sPersonId, int iVersion)
{
     string sUploadFile = sPersonId + ".jpg";

     mClient = (HttpWebRequest)WebRequest.Create("http://" + mHostAddress +
     "/nbws/goforms/upload");
     mClient.Method = "POST";

     byte[] dispositionBuf = Encoding.UTF8.GetBytes("Content-Disposition: form-
     data; name=\"photo\";

     personid=\"" + sPersonId + "\"; filename=\"" + sUploadFile + "\"");
     byte[] typeBuf = Encoding.UTF8.GetBytes("Content-Type: image/jpeg");

     string sPartBoundary = "----------nbapiBoundary" +
     DateTime.Now.Ticks.ToString("x");

     mClient.ContentType = "multipart/form-data; boundary=" + sPartBoundary;
     if ("" != mSessionId)
     {
        mClient.Headers.Add("Cookie", ".sessionId=" + mSessionId);
     }

     byte[] crlfBuf = Encoding.UTF8.GetBytes("\r\n");
```

```
    byte[] terminationBuf = Encoding.UTF8.GetBytes("--" + sPartBoundary + "--
    \r\n");
    byte[] boundaryBuf = Encoding.UTF8.GetBytes(sPartBoundary);

    Stream PostData = mClient.GetRequestStream();

    PostData.Write(Encoding.UTF8.GetBytes("--"), 0, 2);
    PostData.Write(boundaryBuf, 0, boundaryBuf.Length);
    PostData.Write(crlfBuf, 0, crlfBuf.Length);

    PostData.Write(dispositionBuf, 0, dispositionBuf.Length);
    PostData.Write(crlfBuf, 0, crlfBuf.Length);

    PostData.Write(typeBuf, 0, typeBuf.Length);
    PostData.Write(crlfBuf, 0, crlfBuf.Length);
    PostData.Write(crlfBuf, 0, crlfBuf.Length);

    PostData.Write(buffer, 0, buffer.Length);
    PostData.Write(crlfBuf, 0, crlfBuf.Length);

    PostData.Write(terminationBuf, 0, terminationBuf.Length);
    PostData.Close();

    HttpWebResponse response = (HttpWebResponse)mClient.GetResponse();

    Stream responseStream = response.GetResponseStream();

    StreamReader reader = new StreamReader(responseStream);

    mLastResponse = reader.ReadToEnd();

    if (-1 != mLastResponse.IndexOf("<APIERROR>") ||
      -1 != mLastResponse.IndexOf("<CODE>FAIL</CODE>") ||
      -1 != mLastResponse.IndexOf("<CODE>Not Found</CODE>"))
    {
       SetApiError();
       return (false);
    }

    return (true);
}
```

The Event API (NetBox Only)

The API includes an Event API, which can be used by third-party systems to trigger events and receive notification of events (activity).

The Event API commands for retrieving the event list (**ListEvent**) and triggering (**TriggerEvent**) events are accepted at address:

```
http://<NetBox IP Address>/nbws/goforms/nbapi
```

The Event API command for receiving the event stream (**StreamEvent**) is accepted at address:

```
http://<netbox>//nbws/goforms/nbapi
```

Event Triggering

Event definitions are well defined in the NetBox help and *Initial Software Setup Guide*. In the NetBox, you are able to assign Events to activate upon various triggers. The **TriggerEvent** API allows you to activate an Event by triggering it from your application.

Retrieving a List of Events

A program can retrieve the list of configured events by using **ListEvents**, supplying the `sessionid` and using the `sessionid` as a cookie.

Triggering an Event

A program issues a **TriggerEvent** call to activate or deactivate an event.

> **Note**: Activating an event that requires acknowledgement will persist in the system user interface until it is acknowledged.

Streaming Events

With the **StreamEvent** API you can subscribe to receive activity. This available activity is the same as you see in the Activity Log on your NetBox.

Receiving Event Notifications

A program issues a **StreamEvents** call to request streaming of events.

Multiple events can be included in a response. The elements included in the `EVENT` element are dependent upon the specific event type. Not all elements are included in every notification. Event notifications commence starting from the issuance of the request. Past events are not delivered through this interface. The program is responsible for re-issuing the request in the event that it gets disconnected from the system.

Every ten seconds, a keep-alive response containing zero events is streamed to the client via the **StreamEvents** command.

Note: A program is limited to one connection to the NetBox when receiving the event stream.

Filtering Events

A program can choose to limit the events received by specifying filters in a **StreamEvents** command. The filters determine what activity to return based on values specified for any of the event elements.

Event Descriptions

Most Event Descriptions include the tags PARTNAME and CDT. These are not included below for brevity. The available Event Description (DESCNAME) values include additional tags:

Event Description (DESCNAME)	Related Tags
Access Granted	PERSONID, PERSONNAME, PORTALNAME, RDRNAME, ACNAME, ACNUM, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Invalid Access	DETAIL, PERSONID, PERSONNAME, PORTALNAME, RDRNAME, ACNAME, ACNUM, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Portal Held Open	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Portal Forced Open	PORTALNAME, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Portal Restored	PORTALNAME, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Controller Startup	
Network Node Startup	
Network Controller Shutdown	
Momentary Unlock	PORTALNAME, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Unlock	PORTALNAME, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Relock	PORTALNAME, NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Timeout	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Restored	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Disconnect	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Bad Configuration	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Connected	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Identification	DETAIL, NODEADDRESS, NODENAME, NODEUNIQUE, REVISION
Network Node Data Disconnect	

Event Description (DESCNAME)	Related Tags
Log Archive Success	
Log Archive Failure	
Logged In	PERSONID, PERSONNAME, LOGINADDRESS
Logged Out	PERSONID, PERSONNAME, LOGINADDRESS
Login Failed	DETAIL, PERSONID, PERSONNAME, LOGINADDRESS
Request Momentary Unlock	PERSONID, PERSONNAME, PORTALNAME, NODEADDRESS, NODENAME, NODEUNIQUE
Session Expired	PERSONID, PERSONNAME, LOGINADDRESS
Event Triggered	
Event Normal	EVTNAME, EVTPRIO, NDT
Event Activated	EVTNAME, EVTPRIO, NDT
Event Trouble	
Network Node Tamper Alarm	
Network Node DHCP Failed	
Elevator Access Granted	PERSONID, PERSONNAME, RDRNAME, UCBITLENGTH, NDT, NODEADDRESS, NODENAME
Elevator Access Denied	DETAIL, PERSONID, PERSONNAME, RDRNAME, UCBITLENGTH, NDT, NODEADDRESS, NODENAME
Threat Level Set	PERSONID, PERSONNAME, LOCATIONNAME, THREATNAME
Threat Level Set (API)	
Threat Level Set (ALM)	
License Read Failure	
FTP Backup Complete	
FTP Backup Failed	
Alarm Panel Arm Request	ALARMPANELNAME
Alarm Panel Disarm Request	ALARMPANELNAM
Alarm Panel Armed	ALARMPANELNAME
Alarm Panel Disarmed	ALARMPANELNAME
Alarm Panel Arm Failure	ALARMPANELNAME
Alarm Panel Disarm Failure	ALARMPANELNAME
Alarm Panel Arm Interrupted	ALARMPANELNAME

Event Description (DESCNAME)	Related Tags
Network Node Blade Not Responding	NDT, NODEADDRESS, NODENAME, NODEUNIQUE, BLADESLOT
Network Node Blade Responding	NDT, NODEADDRESS, NODENAME, NODEUNIQUE, BLADESLOT
Network Node Coprocessor Not Responding	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Coprocessor Responding	NDT, NODEADDRESS, NODENAME, NODEUNIQUE
NAS Backup Complete	
NAS Backup Failed	
Event Acknowledged	EVTNAME, EVTPRIO, PERSONID, PERSONNAME,
Event Actions Cleared	EVTNAME, PERSONID, PERSONNAME
Access Not Completed	PERSONID, PERSONNAME, PORTALNAME, RDRNAME
Duty Log Entry	PERSONID, PERSONNAME
Battery Voltage Low	
Battery Failed	
Battery Replaced	
Access Denied Because Radio Busy	
Network Node Discovered	
Network Node Configuration and Card Info Reloaded	
Intrusion Panel Connected	IPANELNAME
Intrusion Panel Not Connected	IPANELNAME
Intrusion Panel Request Arm Area	IPANELAREA, IPANELNAME
Intrusion Panel Request Disarm Area	IPANELAREA, IPANELNAME, IPANELUSE
Intrusion Panel Request Bypass Zone	IPANELAREA, IPANELNAME, IPANELUSER, IPANELZONE
Intrusion Panel Request Reset Bypass	IPANELAREA, IPANELNAME, IPANELUSER, IPANELZONE
Intrusion Panel Area Armed	IPANELAREA, IPANELNAME
Intrusion Panel Area Disarmed	IPANELAREA, IPANELNAME
Intrusion Panel Alarm	IPANELAREA, IPANELNAME, IPANELZONE,
Intrusion Panel Restored	IPANELNAM
Intrusion Panel Connection Restored	IPANELNAME
Intrusion Panel Zone Bypassed	IPANELAREA, IPANELNAME, IPANELZONE
Intrusion Panel Zone Reset	IPANELAREA, IPANELNAME, IPANELZONE

Event Description (DESCNAME)	Related Tags
Intrusion Panel Area Late to Alarm	IPANELAREA, IPANELNAME
Intrusion Panel Zone Trouble	IPANELAREA, IPANELNAME, IPANELZONE
Intrusion Panel Zone Fault	IPANELAREA, IPANELNAME, IPANELZONE
Intrusion Panel Zone Restored	IPANELAREA, IPANELNAME, IPANELZONE
Intrusion Panel Request Toggle Output	DETAIL, IPANELAREA, IPANELNAME, IPANELOUTPUT, IPANELUSER
Intrusion Panel Output Toggled	DETAIL, IPANELNAME, IPANELOUTPUT
Intrusion Panel Communication Path Trouble	DETAIL, IPANELNAME
Intrusion Panel Communication Path Restored	IPANELNAME
System Backup Started	
System Backup In Progress	DETAIL
System Backup Successful	
System Backup Failed	
Video Event	DETAIL
Cause Inactive	
Keypad Timed Unlock Expired	
Temporary Credential Issued	
Temporary Credential Returned	
Request Persistent Unlock	PERSONID, PERSONNAME, PORTALNAME, NODEADDRESS, NODENAME, NODEUNIQUE
Request Persistent Lock	PERSONID, PERSONNAME, PORTALNAME, NODEADDRESS, NODENAME, NODEUNIQUE
Request Disable Portal	
Request Enable Portal	
Portal Disabled	
Portal Enabled	
Keypad Command Executed	EVTNAME, RDRNAME
Reader Tamper Alarm	
Reader Tamper Normal	
Reader Battery Alarm	
Reader Battery Normal	
Blade Tamper Alarm	
Blade Tamper Normal	

Event Description (DESCNAME)	Related Tags
Manual Key Override	
Evacuation	FILENAME
Mustering For Evacuation	LOGINADDRESS, FILENAME
System Health	
Reader Communication Alarm	
Reader Communication Normal	
FTP Backup Failed: FTP is Configured and Enabled	
NAS Backup Failed: FTP is Configured and Enabled	
Backup Successfully Copied to FTP Server	
Backup Successfully Copied to NAS Server	
Elevator Free Access	PERSONID, PERSONNAME, NDT, NODEADDRESS, NODENAME
Elevator Access Not Completed	PERSONID, PERSONNAME, RDRNAME, NDT, NODEADDRESS, NODENAME
Emergency Call Activated for Elevator	NDT, NODEADDRESS, NODENAME
Emergency Call Restored for Elevator	NDT, NODEADDRESS, NODENAME
Privacy Enabled	
Interior Push Button	
Door Bolted	
System License Expires in 60 Days	
System License Expires in 30 Days	
System License Expired	
System License Expired Acknowledged	
Network Node System License Not Detected	
Network Node System License Not Detected for 21 Days	
Network Node System License Not Detected for 30 Days	
Network Node System License Reestablished	
Network Node System License Warning Acknowledged	
Network Node System License Error Acknowledged	
Network Controller Takeover	

Event Description (DESCNAME)	Related Tags
Network Controller Configured As Primary	
Network Controller Configured As Standby	
Network Node Secure Configuration Error	DETAIL, NODEADDRESS, NODENAME, NODEUNIQUE
Network Node Secure Communication Failed	DETAIL, NODEADDRESS, NODENAME, NODEUNIQUE

Event Element Descriptions

If the event elements listed in the following table are included in a request, they will appear in responses returned from the system.

You are required to explicitly specify the event elements you want your applications to receive.

The event elements that "support filters" can include a filter specification to restrict activity based on values of the event elements.

Event Element (TAGNAMES)	Supports Filters?	Description
ACNAME	Yes	Access Card Hot Stamp
ACNUM	Yes	Access Card Encoded Number
ACTIVITYID	No	Internal identifier for the activity
ALARMPANELNAME	Yes	Configured name of the alarm panel generating the activity
ALARMSTATENAME	Yes	Internal name for the alarm state, such as Active or Escalated
ALARMTIMERNAME	Yes	Internal name for the timer used to determine when the alarm changes state
ALARMTRANSITIONNAME	Yes	Internal name for the transition of the alarm from one state to another, such as Escalated to Urgent
BLADESLOT	Yes	Slot number of the node blade associated with the activity
CDT	No	Controller date/time associated with the activity
DESCNAME	Yes	General description of the activity
DETAIL	Yes	Additional text detail about the activity
EVTNAME	Yes	Configured name of the event generating the activity
EVTPRIO	No	Configured priority number of the event generating the activity
IPANELAREA	Yes	Intrusion panel area associated with the activity
IPANELNAME	Yes	Intrusion panel name associated with the activity
IPANELOUTPUT	Yes	Intrusion panel output associated with the activity
IPANELUSER	Yes	Intrusion panel user (if any) associated with the activity
IPANELZONE	Yes	Intrusion panel zone associated with the activity
LOCATIONNAME	Yes	Configured location associated with the activity

Event Element (TAGNAMES)	Supports Filters?	Description
LOGINADDRESS	Yes	Host (from which a user logged in) associated with the activity
NODEADDRESS	Yes	IP address of the node associated with the activity
NODENAME	Yes	Configured name for the node associated with the activity
NODEUNIQUE	No	Unique identifier for the node associated with the activity
NDT	No	Node date/time associated with the activity
PARTNAME	Yes	Partition name associated with the activity
PERSONNAME	Yes	Configured person name on the NetBox associated with the activity
PERSONID	No	Configured person identifier on the NetBox associated with the activity
PORTALNAME	Yes	Configured name of the portal associated with the activity
RDRNAME	Yes	Configured name of the card reader associated with the activity
THREATNAME	Yes	Name of the threat level associated with the activity
UCBITLENGTH	No	Number of bits in the card format associated with the activity

Intrusion Panel Integration APIs

NetBox API integrators can create a NetBox Intrusion Panel Integration application (NetBox 5.6 and later systems only). When licensed for "Max NBAPI Intrusion Panels," the NetBox will create an Intrusion Panel identifier for panels of type "LenelS2 NBAPI Panel." Integrators should build a continuously running application (a daemon or a service application) to serve as a mediator between its associated intrusion panels and the NetBox. The integration application should:

- Use the HTTP-based REST API provided by the NetBox.
- Communicate with the NetBox as a WebSocket client.

An integration application can support one or more associated intrusion panels, and it interacts with its associated intrusion panels via the panels' unique proprietary network API. The application then communicates with the NetBox using the REST API and WebSocket messages.

The associated intrusion panel data controlled and maintained by the NetBox can be created and updated by the integration application using the NetBox REST API designed for integration applications. The NetBox manipulates the intrusion panel by sending commands to the integration application using WebSocket messages, and it is responsible for manipulating its associated intrusion panel using the low-level communication API provided by the intrusion panel.

Overview of Intrusion Panel Integration Process

1. License NetBox for "Max NBAPI Intrusion Panels."
2. Understand NetBox REST API for Intrusion panel integration.
3. Add "LenelS2 NBAPI Panel" to NetBox using "Intrusion Panels" configuration application.
4. Understand Identifier values for Intrusion Panels and associated resources.
5. Create Intrusion Panel integration application that supports the following features and behaviors:

   a. Login to NetBox and obtain NetBox Session ID.

   b. Properly synchronize time with NetBox.

   c. Create WebSocket connection to WebSocket server running on the NetBox.

   d. Properly account for NetBox Session ID expiration and WebSocket server unavailability.

   e. Create associated Intrusion Panel resources.

   f. Use NetBox REST API to properly subscribe to all intrusion panels supported by integration application.

   g. Update status of Intrusion Panel and associated resources on NetBox.

   h. Handle NetBox command messages from WebSocket connection.

   i. Handle request to unsubscribe panel.

   j. Manage intrusion panel users.

6. REST API documentation for Intrusion Panel Integration.

Licensing NetBox for NBAPI Intrusion Panels

In order to use the NetBox REST API for intrusion panel integration, the NetBox must be licensed for one or more numbers of "MAX NBAPI Intrusion Panels". This number will correspond to the maximum number of intrusion panels of type "LenelS2 NBAPI Panel" that WILL APPEAR in the NetBox "Intrusion Panels" configuration application.

If no "NBAPI Intrusion Panels" licenses are available, then any invocation of NetBox REST API for intrusion panel endpoints will result in an error with returned data like the following exsample:

```
{
  "success" : false,
  "statusCode" : 400,
  "statusPhrase" : "Bad Request",
  "data" : {
    "errorCode" : "NOT_LICENSED",
    "errorMessage" : null,
    "detail" : null
  }
}
```

Understanding NetBox REST API for Intrusion Panel Integration

All REST API endpoints used by integration applications starting with the URL prefix /nbws/intrusionapi are served by the http server provided by the NetBox. For example, if the NetBox is associated with the local server with DNS name netbox.company.local and the login REST API endpoint is /nbws/intrusionapi/login, then the REST API endpoints should be accessed at the full URL address of http://netbox.company.local/nbws/intrusionapi/login. When an endpoint accepts input data, and when it returns data, the data will be of type JSON. All returned data from the intrusion API endpoint take the form of JSON like the example below:

```
{
  "success" : true,
  "statusCode" : 200,
  "statusPhrase" : "OK",
  "data" : {
    "sessionId" :
     "n6BKNJejrnYIGBpvTj9QOPUe25ObTMDmZCC6qqU7BFOrmFN4NHiGkcwLvV0ZtVA8"
  }
}
```

When the "success" attribute is true, it indicates that no errors were encountered by the REST API invocation and the JSON object associated with the "data" attribute represents desired data. From the example above, the JSON object associated with the "data" attribute represents data to be used from a successful login attempt. The schema of the JSON object associated with the "data" attribute will be different for each REST API endpoint. However, outer JSON object that contains the "data" JSON attribute will always have the same JSON schema shown above. This makes error checking for the NetBox REST API consistent for the different endpoints used for intrusion panel integration.

For example, the JSON below represents data returned from an invocation that resulted in an error condition.

```
 {
  "success" : false,
  "statusCode" : 401,
  "statusPhrase" : "Unauthorized",
  "data" : {
    "errorCode" : "NO_API_PERMISSION",
```

```
    "errorMessage" : null,
    "detail" : null
  }
}
```

Note from the above JSON that “success” attribute is false, which indicates an error condition. The JSON object associated with the “data” attribute provides details about the error condition. The information provided by the Intrusion Panel API information will provide more detailed information about the different error codes. Except for the login endpoint, all other REST endpoints will require that the integration application provide a valid NetBox session ID associated with a custom HTTP header attribute named “lenels2-api-session-id” in order to process the API invocation. See Login to NetBox and obtain NetBox Session ID for further information about the login procedures.

Adding the LenelS2 NBAPI Panel to NetBox using the Intrusion Panels Configuration Application

The NetBox Intrusion Panels configuration application should be used to create intrusion panels of type “LenelS2 NBAPI Panel”. For each intrusion panel to be supported by an integration application, one should create an intrusion panel entry in the Intrusion Panels configuration application. The “Identifier” attribute assigned to the intrusion panel will be used by the integration application when it attempts to manipulate data associated with the intrusion panel on the NetBox. The “Identifier” field is a required attribute for intrusion panels of type “LenelS2 NBAPI Panel,” and should be unique for all panels on the NetBox.

Identifier values for Intrusion Panels and associated resources

The Intrusion Panel identifier for panels of type “LenelS2 NBAPI Panel” must be unique for all intrusion panels within the NetBox. For intrusion panel resources such as Outputs, Areas, and Zones, a unique identifier should be used by the integration application to distinguish each resource assigned to each intrusion panel. It is the integration application’s responsibility to create and manage these resources (see Creating the Intrusion Panel integration application resources for more details). The NetBox REST API provides endpoints for creating these resources, and these endpoints are distinguished from each other with respect to the different resources based on these unique identifiers (i.e. the endpoint to manipulate Areas is /nbws/intrusionapi/zone/{panelId}/{areaId} where {panelId} and {areaId} represents these unique identifiers). It is the integration application’s responsibility to properly maintain unique identifiers for the resources assigned to the integration’s associated intrusion panels.

Identifiers for Outputs and Areas should be unique within the scope of their assigned panels. For example, an output might have and identifier of “output1” when assigned to an intrusion panel with an identifier of “panel1”. This means that no other outputs assigned to the “panel1” intrusion panel can have the identifier of “ouput1”. However, the output identifier of “output1” can be used for another output assigned to another intrusion panel like one with the identifier of “panel2”. Zones are associated to both Panels and Areas, but their identifier have to be unique within a panel too. For Zones, the assignment to a different Area is not sufficient for the zones to share identifiers if they exist on the same panel.

Creating the Intrusion Panel integration application

This section describes the features and behaviors of the Intrusion Panel Integration application.

Login to NetBox and obtain NetBox Session ID

As part of the integration application startup procedure, an integration should login to the NetBox using the NetBox’s REST API endpoint for login, /nbws/intrusionapi/login, and using HTTP POST of JSON data for login request. (See REST API Intrusion Panel Integration for data to send.)

After a successful login, you should store the NetBox session ID provided by the value of the "sessionId" attribute from JSON object associated with the “data” attribute. This session ID should be associated with ”lenels2-api-session-id” custom HTTP header attribute. Failure to provide a valid session ID in the HTTP Headers for use with all other NetBox REST API invocations will result in an HTTP 401 error code error response like the example below:

```
{
  "success" : false,
  "statusCode" : 401,
  "statusPhrase" : "Unauthorized",
  "data" : {
    "errorCode" : "INVALID_AUTH_HEADER",
    "errorMessage" : null,
    "detail" : null
  }
}
```

Properly synchronize time with NetBox

It is important that the NetBox and the intrusion panel integrations have synchronized time for proper interaction with intrusion panels and logging of events. The NetBox has a REST API endpoint at the URL /nbws/intrusionapi/systemtime to provide the current system time of the NetBox. Executing HTTP GET on that URL endpoint will provide the current system time in ISO 8601 date time format for date plus hours, minutes, seconds, and a decimal fraction of a second with time zone offsets (YYYY-MM-DDThh:mm:ss.sTZD). Below is sample JSON data that is returned with a successful execution of that API:

```
{
  "success" : true,
  "statusCode" : 200,
  "statusPhrase" : "OK",
  "data" : {
    "localTime" : "2022-07-24T22:39:39.744407-04:00"
  }
}
```

The attribute “localTime” will have the NetBox system time, and the intrusion panel integration should synchronize its time with that value.

Creating WebSocket connection to WebSocket server running on the NetBox

In order to push commands from the NetBox to the integration application, the integration application should also create a web socket connection to the NetBox using the URL (/nbws/intrusionapi/messages) as part of its startup procedures. Do not forget to provide the NetBox session ID with the “lenels2-api-session-id” custom HTTP header attribute as required by all other non-login REST API endpoints. It is with this WebSocket connection, that the NetBox will directly communicate with the integration application.

Properly account for NetBox Session ID expiration and WebSocket server unavailability

A NetBox session ID might expire, so the integration application should account for the necessity to login again to obtain a new NetBox Session ID. In addition, the WebSocket server on the NetBox might become unavailable due to network outages or a restart of the NetBox, so the integration should account for the possibility that its WebSocket connection to the NetBox might become invalid, and the integration application should be able to make a new connection to the NetBox WebSocket server at any time. When obtaining a new session ID, the integration application should close its current WebSocket connection if it is still valid and obtain a new connection based on this new session ID. In addition, panels that were previously subscribed to

should also be subscribed to again based on this new session ID and new WebSocket connection. *See* Using NetBox REST API to properly subscribe to all intrusion panels supported by the integration application to properly subscribe to all intrusion panels supported by integration application.

Creating associated Intrusion Panel resources

An intrusion panel has assigned resources such as Outputs, Areas, and Zones that are maintained by the associated integration application. These resources should also be recorded on the NetBox, so that NetBox administrators can request that they be manipulated by the integration application. In addition, the NetBox can have these resources be manipulated as part of the NetBox's event processing system. For example, a portal event might require an intrusion panel to arm an assigned area. As part of the integration application startup procedures, the application should synchronize its record of these resources with those that currently exist on the NetBox. The integration application should create any resources that it maintains that currently do not exist on the NetBox. In addition, it should also delete any stale data on the NetBox that currently does not exist in the integration application intrusion panel resource state.

Using NetBox REST API to properly subscribe to all intrusion panels supported by the integration application

An integration application should be aware of the Intrusion Panel identifier values of all the intrusion panels it is associated with. These panel identifier values are entered by NetBox administrators as detailed in Licensing NetBox for NBAPI Intrusion Panels. As part of the integration application's startup procedures, it should subscribe to the panels specified by these panel identifier values using the REST API endpoint specified by the URL (/nbws/intrusionapi/subscribe). See REST API Intrusion Panel Integration for details. This should be done after a successful login and creation of the WebSocket connection with the NetBox.

After a successful subscription, the NetBox will send commands to the integration application as WebSocket messages for each panel it has successfully subscribed to. Theses commands are associated with actions the NetBox requires the associated Intrusion Panel to properly execute. *See* Handling NetBox command messages from WebSocket connection for further details on NetBox commands.

REST API endpoints are provided for querying, creating, and deleting Outputs (/nbws/intrsuionapi/output/{panelId}), Areas (/nbws/intrusionapi/area/{panelId}), and Zones (/nbws/intrusionapi/zone/{panelId}/{areaId}). See REST API Intrusion Panel Integration for details. It is the integration application's responsibility to maintain and to create the proper list of identifiers that are associated with each intrusion panel resource.

Updating status of Intrusion Panel and associated resources on NetBox

The integration application might receive status updates from its associated intrusion panels and the intrusion panel's resources based on the communication connection it maintains with its intrusion panels. The integration application should relay these relevant status changes to the NetBox using one of the appropriate four rest endpoints for updating status. The following are the four status associated endpoints:

1. /panelstatus/{panelId}
2. /outputstatus/{panelId}/{outputId}
3. /areastatus/{panelId}/{areaId}
4. /zonestatus/{panelId}/{areaId}/{zoneId}

These status updates relate to the states of the intrusion panel and are important in case the NetBox needs to do event processing based on intrusion panel and intrusion panel resource status changes. For example, the battery status on an intrusion panel or alarm status of an area assigned to an intrusion panel by need to be

impetus of some NetBox action. The JSON data associated with these status updates are detailed in the REST API documentation mentioned in REST API Intrusion Panel Integration.

Handling NetBox command messages from WebSocket connection

The NetBox will sometimes require that associated Intrusion panels manipulates its state or the state of its assigned resources. The requirements to do this might come from direct action of NetBox administrators using the NetBox GUI or through automatic processing of the NetBox event system for events associated with the Intrusion Panel. To manipulate its associated Intrusion Panels, the NetBox will send WebSocket messages to the integration application that has subscribed to the associated Intrusion Panel. The following are samples of these command messages sent from the NetBox to an integration application:

1) *Command to indicate that a paneol it has subscribed to is no longer valid (refer to* Handling request to unsubscribe panel *for details):*

```
{ "command": "UnsubscribePanel", "panelId": "P0001", "userId": "270-132",
"messageId":"1654633216-116426000" }
```

2) *Command to arm an area:*

```
{ "command": "ArmArea", "panelId": "P0001", "areaId": "273", "userId":
"270-132", "messageId":"1654633217-116476000" }
```

3) *Command to disarm an area:*

```
{ "command": "DisarmArea", "panelId": "P0001", "areaId": "273", "userId":
"270-132", "messageId":"1654633226-116427000" }
```

4) *Command to bypass a zone associated with an area:*

```
{ "command": "BypassZone", "panelId": "P0001", "areaId": "273", "zoneId":
"155", "userId": "270-133", "messageId":"1654633236-216406000" }
```

5) *Command to reset a zone as no longer being bypassed in an associated area:*

```
{ "command": "UnbypassZone", "panelId": "P0001", "areaId": "273",
"zoneId": "155", "userId": "270-133", "messageId":"1654633258-316496000" }
```

6) *Command to activate an output:*

```
{ "command": "ActivateOutput", "panelId": "P0001", "outputId": "155",
"userId": "270-133", "messageId":"1654654276-446438000" }
```

7) *Command to deactivate an output:*

```
{ "command": "DeactivateOutput", "panelId": "P0001", "outputId": "155",
"userId": "270-133", "messageId":"1654633806-556445000" }
```

After receiving such WebSocket messages, the integration should log these messages and properly manipulate its associated intrusion panel. After manipulating the intrusion panel, the integration application should update any associated status to the NetBox. It is especially important that the integration application log the associated "messageId" value of the command message and log this "messageId" value with any action it might take due to command messages. The "messageId" value can be used by technical support engineers to trace messages through the system defined by the NetBox and integration application interaction. Tech support will be able to surmise why a command was sent to the integration application from the NetBox by reviewing NetBox logs, and with the logs of the integration application they can trace why an intrusion panel action was taken based on the trace of the "messageId" value.

The following are examples of the actions an integration application will take after receiving one of the six status update-based command messages:

1. If “ArmArea” is received, integration application should tell the associated intrusion panel to arm the specified area and log that this action was taken due to the associated “messageId”. If the area was successfully armed, the “armedStatus” status attribute should be updated to “ARMED” on the NetBox using the /nbws/intrusionapi/areastatus/{panelId}/{areaId} endpoint.
2. If “DisarmArea” is received, integration application should tell the associated intrusion panel to disarm the specified area and log that this action was taken due to the associated “messageId”. If the area was successfully disarmed, the “armedStatus” status attribute should be updated to “NOT_ARMED” on the NetBox using the /nbws/intrusionapi/areastatus/{panelId}/{areaId} endpoint.
3. If “BypassZone” is received, integration application should tell the associated intrusion panel to bypass the specified zone in the specified area and log that this action was taken due to the associated “messageId”. If the zone was successfully bypassed, the “overallStatus” status attribute should be updated to “BYPASSED” on the NetBox using the /nbws/intrusionapi/zonestatus/{panelId}/{areaId}/{zoneId} endpoint.
4. If “UnbypassZone” is received, integration application should tell the associated intrusion panel to reset the specified zone in the specified area and log that this action was taken due to the associated “messageId”. If the zone was successfully reset, the “overallStatus” status attribute should be updated to “NORMAL” on the NetBox using the /nbws/intrusionapi/zonestatus/{panelId}/{areaId}/{zoneId} endpoint.
5. If “ActivateOutput” is received, integration application should tell the associated intrusion panel to activate the specified output and log that this action was taken due to the associated “messageId”. If the output was successfully activated, the “overallStatus” status attribute should be updated to “STEADY” on the NetBox using the /nbws/intrusionapi/outputstatus/{panelId}/{outputId} endpoint.
6. If “DeactivateOutput” is received, integration application should tell the associated intrusion panel to deactivate the specified output and log that this action was taken due to the associated “messageId”. If the output was successfully deactivated, the “overallStatus” status attribute should be updated to “OFF” on the NetBox using the /nbws/intrusionapi/outputstatus/{panelId}/{outputId} endpoint.

For example, an integration application has subscribed to receive websocket command messages for an intrusion panel identified by “PANEL123”. It then receives a command message like the following from the NetBox:

```
{ “command”: “DeactivateOutput”, "panelId": "PANEL123", “outputId”: “OUTPUT1”,
    “userId”: “270-133”, "messageId":"164633806-556445000" }
```

The integration application should then communicate with its associated intrusion panel to execute the necessary proprietary commands to deactivate the output associated with the identifier “OUTPUT1”. The integration application should log in its own log file that it deactivated output “OUTPUT1” due to a command message from the NetBox associated with message identifier “164633806-556445000”. After confirming the output has been deactivated, the integration application should update the output’s status with the NetBox by executing HTTP PUT on the NetBox’s REST API endpoint:

```
/nbws/intrusionapi/outputstatus/PANEL123/OUTPUT1
```

with the following JSON data:

```
{
     “overallStatus” : “OFF”
}
```

Handling request to unsubscribe panel

Sometimes on the NetBox, an existing Intrusion Panel of type “LenelS2 NBAPI Panel” is deleted or might have its “Identifier” value changed. If an integration application is currently subscribed to this existing Intrusion Panel on the relevant NetBox, then the integration application will receive the “UnsubscribePanel” command message on its websocket connection like the example below:

```
{ “command”: “UnsubscribePanel”, "panelId": "P0001", “userId”: “270-132”,
    "messageId":"1654633216-116426000" }
```

In this case, the integration application should consider the associated intrusion panel as being removed from its records. It should take the necessary action of shutting down the panel if its communication API with the panel supports such actions and log the “messageId” of the command that was the impetus for this shutdown action. It will be the integration application administrator’s responsibility to reconfigure the integration application to account for changes in the set the of Intrusion Panels the integration is responsible for managing.

Managing intrusion panel users

Each Intrusion Panel can be associated with a set of intrusion panel users. The integration application should maintain this set of users and their associated unique user identifiers and synchronize this set with the set of intrusion panel users maintained by the NetBox. This should be done at integration application startup and due to any dynamic changes initiated by the integration’s associated intrusion panel. There is a NetBox configuration application that allows these intrusion panel users to be associated with NetBox user accounts. This allows actions taken on the NetBox by these associated user accounts to map to Intrusion Panel user accounts maintained by the integration application. The unique user identifiers intrusion panel users that mapped to NetBox user accounts will appear in the WebSocket command messages, so that they can be properly logged by the integration application handled appropriately by the integrations associated intrusion panel if necessary.

For example, if a mapped NetBox user account is associated with user identifier “USER123” and that NetBox account deactivates an output on a panel identified by “PANEL1”, the integration application will receive the following command message if it is properly subscribed to the associated panel:

```
{ “command”: “DeactivateOutput”, "panelId": "PANEL1", “outputId”: “155”,
“userId”: “USER123”, "messageId":"1654633806-556445000" }
```

The REST endpoints associated with Intrusion Panel user management that the integration application should use are the following:

1. /nbws/intrusionapi/user/{panelId}/{userId}
2. /nbws/intrusionapi/panel/users/{panelId}

The URL (/nbws/intrusionapi/user/{panelId}/{userId}) is used for creating, retrieving, and deleting individual users specified by panel and user identifier using conventional HTTP POST, GET, and DELETE actions. The URL (/nbws/intrusionapi/panel/users/{panelId}) should be used to obtain a list of all users associated with the specified panel. See REST API Intrusion Panel Integration for more details.

REST API Intrusion Panel Integration

This section contains the NetBox REST API documentation for Intrusion Panel Integration related APIs:

Tags	Request	Description
AREA	get get post put del	Retrieve all areas for a panel. Retrieve a specified panel area. Add an area to a panel. Update a panel area. Delete a panel area.
AREASTATUS	get put	Retrieve a panel area status. Update a panel area status.
LOGIN	post	Establishes a session with NetBox, which will be used in subsequent calls.
LOGOUT	get	Log out from NetBox.
OUTPUT	get get post put del	Retrieve all outputs for a panel. Retrieve a panel output. Add a panel output to a panel. Update a panel output. Delete a panel output.
OUTPUTSTATUS	get put	Retrieve a panel output status. Update a panel output status.
PANELSTATUS	get put	Retrieve a panel status. Update a panel status.
PANELUSER	get get post put del	Get all panel usernames for a specific panel from NetBox. Update a panel username on NetBox. Add a panel username to NetBox. Update a panel username on NetBox. Delete a panel username from NetBox.
SUBSCRIBE	post post	Subscribe for panel notification messages. Subscribe for panel notification messages for a single panel.
SYSTEMTIME	get	Retrieve system time of NetBox.
UNSUBSCRIBE	del	Unsubscribe panel from notification messages.
ZONE	get get post put del	Retrieve all zones for a panel. Retrieve a panel zone. Add a panel zone to a panel and area. Update a panel zone. Delete a panel zone.
ZONESTATUS	get put	Retrieve a panel zone status. Update a panel zone status.

Using the API with Global

The API has a subset of commands that permit network-connected systems to perform various operations with a Global system under program control. For the most part, command usage differs only in the HTTP message address and how partitions are managed.

HTTP Message Address

The API for the commands on Global is invoked by posting an HTTP message to the web server on Global:

```
http://<Global IP address>/global/goforms/nbapi
```

The API on Global is always enabled. Unlike API on NetBox, you cannot disable API on Global.

Authentication

MAC authentication is not supported.

User Roles

When the application logs into Global with the API, the user specified in the **Login** command must have the Global Setup role.

Global API Commands

API on Global supports the following commands with differences from NetBox as noted:

- **Login/Logout**
- **GetPartitions**: Includes the following additional command response elements:
  - `NETBOXNAME`: Name of the NetBox to which `PARTITIONKEY` corresponds.
  - `NETBOXIPADDRESS`: IP address of the NetBox to which `PARTITIONKEY` corresponds.
- **SwitchPartition**: Includes an additional `PARTITIONKEY` value of 0 which specifies that the current login session, initiated by a **Login** call, has no partition assigned. 0 is the default value.

  After a **Login** call is issued, the partition remains unassigned until a **SwitchPartition** call is issued with a `PARTITIONKEY` value.

  A **SwitchPartition** call requires a `PARTITIONKEY` parameter. The `PARTITIONKEY` value corresponds to a partition defined in the Global database. After a **SwitchPartition** call is issued, all commands apply to that partition.

  When the last **SwitchPartition** call contains a `PARTITIONKEY` value of 0 (no partition assigned), certain commands will calls require a `PARTITIONKEY` parameter to be included in the call to specify the partition, such as **AddPerson**.
- **SearchPersonData**: Includes an additional calling parameter:

  - PARTITIONKEY: Specifies the partition to which the **SearchPersonData** call applies, because there can be NetBox systems with duplicate partition names defined in the Global database.

    If the PARTITIONKEY value has been specified with a value other than 0 in the last **SwitchPartition** call, a **SearchPersonData** call will only return the person record data for the partition with the assigned key.

    If the PARTITIONKEY specified in the last **SwitchPartition** call has a value of 0, the **SearchPersonData** call defaults to ALLPARTITIONS.

    If the value for ALLPARTITIONS is TRUE and a PARTITIONKEY is specified in the last **SwitchPartition** call, all partitions are searched for with the specified PARTITIONKEY.

- **GetPerson**: Includes an additional calling parameter:
  - PARTITIONKEY: Required to specify the partition to which the **GetPerson** call applies.

    The person with the person record with the specified PERSONID is retrieved from the Global database, even if deleted (disabled).

- **AddPerson** and **ModifyPerson**: Include an additional calling parameter:
  - PARTITIONKEY: When the last **SwitchPartition** call contains a PARTITIONKEY value of 0 (no partition assigned), subsequent **AddPerson** or **ModifyPerson** calls require a PARTITIONKEY parameter to be included in the call to specify the partition.

    If the last **SwitchPartition** call includes a PARTITIONKEY with a value other than 0, which means that it corresponds to a specific partition, the PARTITIONKEY must not be included in the call. **AddPerson** and **ModifyPerson** modifications will be carried out on the partition key specified in the **SwitchPartition** call until a new **SwitchPartition** call is issued.

- **GetAccessLevels**: Includes an additional calling parameter:
  - PARTITIONKEY:

    Required if the PARTITIONKEY in the last **SwitchPartition** command has a value of 0 (no assigned partition).

    Not required if the PARTITIONKEY in the last **SwitchPartition** call has a value other than 0 (an assigned partition). The **GetAccessLevels** call returns only the access levels for the assigned partition.

- **GetAccessLevels**: Output includes the element PARTITIONKEY to distinguish where the access levels are from. The output ACCESSLEVEL element also includes NAME.
- **GetAccessLevels**: Does not support the element WANTKEY to request Access Level key values vs. Access Level names.
- **GetCardFormats**
- **AddCredential** and **RemoveCredential** commands do not support the following calling parameters:
  - CARDSTATUS
  - CARDEXPDATE

Global NBAPI Partition Management

Command usage for Global differs from NetBox in that partition lists include the system the partition belongs to. In addition, Global can operate with partitions in one of two ways:

- Explicitly specifying partitions in API commands.
- Using the **SwitchPartition** command as is done with a NetBox.

NetBox

For the NetBox API, when a **Login** call is issued, the current partition defaults to the Master partition which has a `PARTITIONKEY` value of 1. The current partition can be changed by issuing a **SwitchPartition** call with a `PARTITIONKEY` value. Other API calls are dependent on this setting.

For example, the following **GetPartitions** call requests the list of partition definitions in the system.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetPartitions" num="1">
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetPartitions** call returns a list of partition definitions.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPartitions" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PARTITIONS>
                <PARTITION>
                    <PARTITIONKEY>1</PARTITIONKEY>
                    <NAME>Master</NAME>
                    <DESCRIPTION>The default partition</DESCRIPTION>
                </PARTITION>
                <PARTITION>
                    <PARTITIONKEY>2</PARTITIONKEY>
                    <NAME>Second Partition</NAME>
                    <DESCRIPTION></DESCRIPTION>
                </PARTITION>
                <PARTITION>
                    <PARTITIONKEY>3</PARTITIONKEY>
                    <NAME>Third Partition</NAME>
                    <DESCRIPTION></DESCRIPTION>
                </PARTITION>
            </PARTITIONS>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

A **SwitchPartition** call changes the current partition to the partition corresponding to PARTITIONKEY 3.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="SwitchPartition" num="1">
       <PARAMS>
          <PARTITIONKEY>3</PARTITIONKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Global

For the Global API, when a **Login** call is issued, the current partition defaults to a PARTITIONKEY value of 0, which specifies that the partition is **unassigned**. Subsequent commands require that a PARTITIONKEY is specified to identify the partition to which the commands will apply.

For the Global API, a **GetPartitions** call also returns a NETBOXNAME and NETBOXIPADDRESS in the response, which identifies the NetBox on which each partition resides.

The **GetPartitions** call, in the example below, shows two NetBox systems that have the same Master partition name with different IP addresses.

```
<NETBOX sessionid="1626934585">
  <RESPONSE command="GetPartitions" num="1">
    <DETAILS>
      <PARTITIONS>
        <PARTITION>
          <PARTITIONKEY>10</PARTITIONKEY>
            <NAME>Master</NAME>
            <DESCRIPTION>The default partition</DESCRIPTION>
            <NETBOXNAME>nb24</NETBOXNAME>
            <NETBOXIPADDRESS>10.0.0.24</NETBOXIPADDRESS>
          </PARTITION>
          <PARTITION>
            <PARTITIONKEY>11</PARTITIONKEY>
              <NAME>Master</NAME>
              <DESCRIPTION>The default partition</DESCRIPTION>
              <NETBOXNAME>nb41</NETBOXNAME>
              <NETBOXIPADDRESS>10.0.0.41</NETBOXIPADDRESS>
           </PARTITION>
         </PARTITIONS>
    </DETAILS>
    <CODE>SUCCESS</CODE>
  </RESPONSE>
</NETBOX>
```

At initial login, your session is not pointing to any partition. It remains unassigned until a **SwitchPartition** call is issued which specifies a PARTITIONKEY. A **SwitchPartition** call issued with PARTITIONKEY value of 10 will change the partition to "Master" on the NetBox named, "nb24." All subsequent commands will apply to the Master partition on "nb24."

A **SwitchPartition** call issued with PARTITIONKEY value of 11 will change the partition to "Master" on the NetBox named, "nb41." All subsequent commands will apply to the Master partition on "nb24."

A call to **SwitchPartition** with a PARTITIONKEY value of 0 will change the partition to **unassigned**.

Command Reference

The API commands are categorized by operation type below. Some commands are listed for multiple categories.

Actions

- *Activate Output*
- *Deactivate Output*
- *DogOnNextExitPortal*
- *LockPortal*
- *MomentaryUnlockPortal*
- *SetThreatLevel*
- *UnlockPortal*

Configuration

- *AddAccessLevel*
- *AddAccessLevelGroup*
- *AddHoliday*
- *AddPartition*
- *AddPortalGroup*
- *AddReaderGroup*
- *AddTimeSpec*
- *AddTimeSpecGroup*
- *AddThreatLevel*
- *AddThreatLevelGroup*
- *DeleteAccessLevel*
- *DeleteAccessLevelGroup*
- *DeleteHoliday*
- *DeletePortalGroup*
- *DeleteReaderGroup*
- *DeleteTimeSpec*
- *GetAccessLevel*
- *GetAccessLevels*
- *GetAccessLevelGroup*
- *GetAccessLevelGroups*
- *GetAccessLevelNames*
- *GetCardFormats*
- *GetElevators*
- *GetFloors*
- *GetHoliday*
- *GetHolidays*
- *GetOutputs*
- *GetPartitions*
- *GetPortalGroup*
- *GetPortalGroups*
- *GetReaderGroup*
- *GetReaderGroups*
- *GetReaders*
- *GetTimeSpecGroup*
- *GetTimeSpecGroups*
- *GetTimeSpecs*
- *GetUDFLists*
- *GetUDFListItems*
- *ModifyAccessLevelGroup*
- *ModifyHoliday*
- *ModifyPortalGroup*
- *ModifyReaderGroup*
- *ModifyThreatLevel*
- *ModifyThreatLevelGroup*
- *ModifyTimeSpec*
- *ModifyTimeSpecGroup*
- *ModifyUDFListItems*
- *RemoveThreatLevel*
- *RemoveThreatLevelGroup*
- *SetThreatLevel*

Events

- *ListEvents*
- *StreamEvents*
- *TriggerEvent*

History

- *GetEventHistory*
- *GetCardAccessDetails*

People

- *AddAccessLevelGroup*
- *AddCredential*
- *AddPerson*
- *GetAccessLevelNames*
- *GetPerson*
- *GetPicture*
- *ModifyAccessLevel*
- *ModifyCredential*
- *ModifyPerson*
- *RemoveCredential*
- *RemovePerson*
- *SearchPersonData*

Portals

- *AddPortalGroup*
- *AddReaderGroup*
- *DeletePortalGroup*
- *DeleteReaderGroup*
- *GetPortalGroup*
- *GetPortalGroups*
- *GetReader*
- *GetReaders*

GetReaderGroup
GetReaderGroups
ModifyPortalGroup
ModifyReaderGroup

Threat Levels

AddThreatLevel
AddThreatLevelGroup
ModifyThreatLevel
ModifyThreatLevelGroup
SetThreatLevel

Utility

GetAPIVersion
GetPartitions
Login
Logout
PingApp
SwitchPartition

ActivateOutput

The **ActivateOutput** command requests the output specified by OUTPUTKEY to activate.

Use the **Deactivate Output** command to request to deactivate an output.

Calling Parameters

- OUTPUTKEY: Key associated with the output to activate.

  Use **GetOutputs** to retrieve the list of OUTPUTKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the OUTPUTKEY associated with the output activation.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Invalid output key
  - Missing OUTPUTKEY
  - Output not online or unreachable
  - Output state not changed.
  - SQL function returned unexpected value

Example

ActivateOutput call to activate the output associated with an OUTPUTKEY with a value of 36.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ActivateOutput" num="1">
       <PARAMS>
          <OUTPUTKEY>36</OUTPUTKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **ActivateOutput** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ActivateOutput" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <OUTPUTKEY>36</OUTPUTKEY>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

AddAccessLevel

The **AddAccessLevel** command creates a new access level.

Calling Parameters

- ACCESSLEVELNAME: The access level name of up to 64 characters.
- ACCESSLEVELDESCRIPTION: The access level description.
- READERKEY (optional): Key corresponding to a reader assigned to this access level.
- READERGROUPKEY (optional): Key corresponding to a reader group assigned to this access level.
- TIMESPECGROUPKEY (required): Key corresponding to the time spec group assigned to this access level.
- THREATLEVELGROUPKEY (optional): Key corresponding to the threat level group assigned to this access level.

Use **GetReaders**, **GetReaderGroups**, or **GetTimeSpecGroups** to retrieve the list of READERKEY, READERGROUPKEY, or TIMESPECGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the ACCESSLEVELKEY in the DETAIL element block as the identifier for the new access level.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Name cannot be empty.
  - Access Level Name exceeds 64
  - For readers, both individual and group cannot have value.

Example

AddAccessLevel call to add the new access level "Cleaning Crew:"

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="AddAccessLevel" num="1">
    <PARAMS>
       <ACCESSLEVELNAME>Cleaning Crew</ACCESSLEVELNAME>
       <ACCESSLEVELDESCRIPTION>4PM to 11PM</ACCESSLEVELDESCRIPTION>
       <READERGROUPKEY>21</READERGROUPKEY>
       <TIMESPECGROUPKEY>21</TIMESPECGROUPKEY>
       <THREATLEVELGROUPKEY>42</THREATLEVELGROUPKEY>
    </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **AddAccessLevel** call indicates that the new access level was added with an ACCESSLEVELKEY value of 10:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddAccessLevel" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <ACCESSLEVELKEY>10</ACCESSLEVELKEY>
           </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddAccessLevelGroup

The **AddAccessLevelGroup** command creates a new access level group.

Calling Parameters

- NAME: Name for access level group of up to 64 characters.
- DESCRIPTION: Description of the new access level group.
- (Optional) Either PARTITIONKEY or SYSTEMGROUP may be supplied:
  - PARTITIONKEY: Partition of the access level group; if not supplied and SYSTEMGROUP not set, defaults to the current partition.
  - SYSTEMGROUP: With a value of '1', creates a multi-partition group in the Master Partition. If the SYSTEMGROUP is set and PARTITIONKEY is empty, then the default partition key 1 is used by the api.
- ACCESSLEVELS: List of access levels to be included in the group. Each ACCESSLEVEL element block includes:
  - Either NAME or KEY
    - NAME: Access level name for the access level.
    - KEY: Access level key for the access level.
  - PARTITIONKEY (required for multi-partition access level groups): Partition of the access level. level. If the value is not passed in, then the PARTITIONKEY value of the group is used for the access level partition key.

Use **GetAccessLevelGroups** to retrieve the list of ACCESSLEVELGROUPKEY values. Use **GetAccessLevelNames** to retrieve the full list of access level names.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the ACCESSLEVELGROUPKEY in the DETAIL element block as the identifier for the new access level group
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Access Level Group Name is a required field
  - Access Level Group Name exceeds max size of 64 characters
  - Duplicate access level group name for specified partition
  - For a single partition group, access level partition id param is optional, if passed in, it should be same as access level group partition id.
  - Access Level [id= XX] does not exist in Access Level Group partition XX
  - Access Level [Name=XX] does not exist in alg partition XX
  - Unable to add or modify access level group: invalid access level parameter(s)
  - Invalid PARTITIONKEY value, must be greater than 0

- Access Level [Name=XX] or [Id=XX] does not exist in alg partition XX
- Not a number: XXX

Example

AddAccessLevelGroup call to add the new single-partition access level group "Developers" using access level keys.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="AddAccessLevelGroup" num="1">
        <PARAMS>
            <NAME>Developers</NAME>
            <DESCRIPTION>Access for all developers</DESCRIPTION>
            <PARTITIONKEY>4</PARTITIONKEY>
            <ACCESSLEVELS>
                <ACCESSLEVEL><KEY>59</KEY></ACCESSLEVEL>
                <ACCESSLEVEL><KEY>60</KEY></ACCESSLEVEL>
            </ACCESSLEVELS>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **AddAccessLevelGroup** call indicates that the new access level group was added with an `ACCESSLEVELGROUPKEY` value of 75:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddAccessLevelGroup" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
           <ACCESSLEVELGROUPKEY>75</ACCESSLEVELGROUPKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddAccessLevelGroup call to add the new multi-partition access level group "Developers" using access level names and specifying partition key.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="AddAccessLevelGroup" num="1">
        <PARAMS>
            <NAME>Developers</NAME>
            <DESCRIPTION>Access for all developers</DESCRIPTION>
            <SYSTEMGROUP>1</SYSTEMGROUP>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <NAME>Main Doors</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>Laboratory</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

AddCredential

The **AddCredential** command adds a credential to a person record in the system database.

Calling Parameters

- PERSONID: ID number of the person associated with the credential to be added.
- CARDFORMAT: Name of the format to be used to decode the credential.
- Either or both of the following parameters must be included in the XML command:
  - ENCODEDNUM : The encoded number for the credential. If this is not supplied, the HOTSTAMP value is assigned to the ENCODEDNUM parameter.
  - HOTSTAMP: The hotstamp number for the credential. If this is not supplied, the ENCODEDNUM value is assigned to the HOTSTAMP parameter.
- WANTCREDENTIALID: Used to request that the system return the credential ID for the credential being added. The credential ID is an alias for the actual credential number.

  As a security measure, credential IDs can be retrieved and stored in a client system in place of the encoded numbers and/or hotstamp numbers. This will allow people to manage credentials from the client system without seeing the actual credential numbers.
- CARDSTATUS: Text string that specifies the status of the credential. If a CARDSTATUS is not included in the command, the default status ACTIVE is assigned to the CARDSTATUS parameter.

  This parameter can be included only if the DISABLED parameter is not included in the call.

Note: An administrator can use the *Credential Attributes* page (under *Configuration : Access Control in the NetBox web interface*) to rename nine of the available credential status settings.

- CARDEXPDATE: Expiration date for the credential. See *Date Formats*.

  A valid Credential Expiration date must be passed as a parameter if *Enable Credential Expiration Requirement* is selected on the Access Control tab of the Network Controller page.

NOTE: CARDSTATUS AND CARDEXPDATE ARE NOT SUPPORTED FOR THE GLOBAL API.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the ACCESSLEVELGROUPKEY in the DETAIL element block as the identifier for the new access level group

- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block
  - Credential expiration date is required
  - Error, Card Status name is either invalid or not allowed
  - Cardformat must be enabled in order to add credential
  - Person NOT FOUND
  - CARDFORMAT NOT FOUND.
  - INVALID PERSONID
  - The card number is already in use.
  - Must supply Encoded Number or Hotstamp.
  - Must supply Person ID and Card Format.
  - Length of STRAC 128 bit Encoded number must be 32 digits
  - Length of STRAC 128 bit HotStamp must be 32 digits
  - Length of FIPS 201 75-bit Encoded number must be 14 digits
  - Length of FIPS 201 75-bit HotStamp must be 14 digits
  - Length of FIPS 128 bit Encoded number must be 32 digits
  - Length of FIPS 128 bit HotStamp must be 32 digits

Example

AddCredential call to request that a credential with the Hotstamp and Encoded number 1111 be added to a person with an ID number of 1001. The required card format is "26 bit Wiegand." The call includes a request for the credential ID:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="AddCredential" num="1">
        <PARAMS>
              <PERSONID>1001</PERSONID>
              <CARDFORMAT>26 bit Wiegand</CARDFORMAT>
              <HOTSTAMP>1111</HOTSTAMP>
              <ENCODEDNUM>1111</ENCODEDNUM>
              <WANTCREDENTIALID>1</WANTCREDENTIALID>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **AddCredential** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddCredential" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
           <CREDENTIALID>14</CREDENTIALID>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddHoliday

The **AddHoliday** command adds a holiday to the system.

There is a limit of 30 holidays that can be defined per partition.

Calling Parameters

- HOLIDAYNAME: Holiday name of up to 64 characters.
- HOLIDAYGROUPS: Specify any or all of the holiday group values (1, 2, 3, 4, 5, 6, 7, 8 separated by commas).
- STARTDATE: Holiday start date. See *Date Formats*.
- ENDDATE: Holiday end date. See *Date Formats*.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the HOLIDAYKEY as the identifier for the added holiday in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block
  - Name already exists.
  - STARTDATE and ENDDATE must be specified in YYYY-MM-DD HH:MM or YYYY-MM-DD HH:MM:SS format.
  - A field value with a maximum of 64 characters was too long.
  - A partition can hold maximum of 30 holidays

Example

AddHoliday call to add a holiday:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="AddHoliday" num="1">
       <PARAMS>
          <HOLIDAYNAME>Christmas</HOLIDAYNAME>
          <HOLIDAYGROUPS>1</HOLIDAYGROUPS>
          <STARTDATE>2016-12-25 00:00</STARTDATE>
          <ENDDATE>2016-12-26 00:00</ENDDATE>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **AddHoliday** call returns a HOLIDAYKEY value of 1:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="AddHoliday" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <HOLIDAYKEY>1</HOLIDAYKEY>
       </DETAILS>
    </RESPONSE>
```

```
</NETBOX>
```

AddPartition

The **AddPartition** command adds a partition. The command requires full system setup privilege.

Calling Parameters

- NAME: Name of the new partition.
- DESCRIPTION: Description of the new partition.
- TIMEZONE: Time zone of the new partition. The list of valid time zones is presented on the *Network Controller Time Settings* page (*Configuration : Time : Network Controller*).

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PARTITIONKEY as the identifier for the new partition in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block
  - Name cannot be empty
  - Name already exists
  - Timezone cannot be empty
  - Invalid timezone

Example

AddPartition call to add a partition:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="AddPartition" num="1">
        <PARAMS>
           <NAME>Sales Demo</NAME>
           <DESCRIPTION>Sales Demo Northeast Region</DESCRIPTION>
           <TIMEZONE>America/New York</TIMEZONE>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **AddPartition** call returns a PARTITIONKEY value of 4:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddPartition" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
           <PARTITIONKEY>4</PARTITIONKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddPerson

The **AddPerson** command allows you to add a new person record. An error is returned if a person record with the same PERSONID in the **AddPerson** call already exists.

If no PERSONID is supplied in the **AddPerson** call, the API creates one for the person record.

The person's record being added can support the following methods of login:

- Traditional username and password login (DB)
- LDAP password verification (LDAP)
- Okta OpenID Connect login (SSO)

Calling Parameters

- PERSONID: ID of the person for whom the credential is to be added.

  If PERSONID matches an existing person, the person record is not created. If no PERSONID is supplied, one is created, and is returned in the result.
- FIRSTNAME: First name of person being added.
- MIDDLENAME: Middle name of person being added.
- LASTNAME: Last name of person being added. This is a required field.
- USERNAME: Username that will be associated with a person's NetBox login account. This is a required field.
- PASSWORD: Password associated with NetBox login account. This field should not be set if the AUTHTYPE is SSO or LDAP.

  If the AUTHTYPE is SSO or LDAP, then PASSWORD is not applicable and should not be set. If AUTHTYPE is DB, then PASSWORD is required.
- ROLE: Role for a person's record when they Login to the NetBox. This is a required field.
- AUTHTYPE: Authorization type for a person's record. Valid values are DB, LDAP, or SSO. This is a required field.
- MOBILEPHONE: Mobile phone number.
- MSUENABLED: Enables Mobile Security User (MSU) mobile credentials. Email address (CONTACTEMAIL) or mobile number (MOBILEPHONE) are required to enable MSU.
- BLUEDIAMONDENABLED: Enables BlueDiamond mobile credentials.
- NOTES: Notes field of person record.
- EXPDATE: Expiration date for person record. See *Date Formats*.
- ACTDATE: Activation date for person record. See *Date Formats*.
- UDF1...UDF20: User-defined fields (20).

- PIN: PIN number which may be required for a card reader with a numeric pad.
- ACCESSLEVELS: Element block containing one or more access levels (maximum of 32) to be associated with the person. Access levels in excess of the maximum will be silently ignored. For an access level to be added to a person record, it must already have been defined in your NetBox. Otherwise, the command will fail. The ACCESSLEVELS element block contains a separate ACCESSLEVEL element for each access level.

  There are two forms of syntax: Access Levels by Name only and Access Levels by Name and Flag. See *Person Access Levels* for a detailed explanation.

  The first form of syntax is "Access Levels by Name only". The ACCESSLEVELS element block contains a single element for each ACCESSLEVEL.

  - ACCESSLEVEL: Access level name.

    The second form of syntax is "Access Levels by Name and flag" and may be referred to as "Alternative Syntax." The ACCESSLEVELS element block contains multiple elements for each ACCESSLEVEL:

  - ACCESSLEVEL: Access level element block
  - ACCESSLEVELNAME: Access level name.
  - DELETE: A value of 1 will result in an access level being removed from the person record. A value of 0 will result in an access level being added to the person record.
  - ACTDATE: NO VALUE MEANS NOW. A VALID DATE STRING MEANS SET THE DATE TO THE VALUE PASSED IN. EXCLUSION OF THE PARAMETER MEANS NOW. NOTE THAT THIS VALUE IS NOT ENFORCED, ONLY RECORDED.
  - EXPDATE: NO VALUE MEANS NEVER. A VALID DATE STRING MEANS SET THE DATE TO THE VALUE PASSED IN. EXCLUSION OF THE PARAMETER MEANS NEVER.
  - AUTOREMOVE: A VALUE OF 1 MEANS THE ACCESS LEVEL WILL BE REMOVED FROM THE PERSON RECORD WHEN THE ACCESS LEVEL EXPIRES.
- PICTURE: (optional) Picture data for the person being added. This data must be Base64 encoded and the file size must not exceed 650KB. (Note that the maximum size returned in the GetPicture commands is 120KB.)
- PICTUREEXT: (optional) Extension that describes the format of the picture data (for example, "jpg"). If not specified, the API will assign an extension of "gif".
- PICTUREURL: (optional): Name of file for picture data as it will be stored on the system. If not specified, the API will assign a filename with the format: lastname firstname.extension
- BADGELAYOUT: Name of the photo ID badging layout file.
- CONTACTPHONE: Office phone number.

- CONTACTEMAIL: Office email address.
- CONTACTSMSEMAIL: Office SMS email address.
- CONTACTLOCATION: Office location.
- OTHERCONTACTNAME: Emergency contact name.
- OTHERCONTACTPHONE1: Emergency contact phone number.
- OTHERCONTACTPHONE2: Alternate emergency contact phone number.
- VEHICLES: The VEHICLES element block contains a set of VEHICLE elements for each vehicle:
  - VEHICLECOLOR
  - VEHICLEMAKE
  - VEHICLEMODEL
  - VEHICLESTATE
  - VEHICLELICNUM
  - VEHICLETAGNUM
- PARTITIONKEY (Global API Only): The partition to which the **AddPerson** call applies to when the application has not set a partition with **SwitchPartition**
Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates that the person record has been successfully added and may return additional elements and element blocks in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Could not find prototype for nbapi.GetInsertIdFromPerson – internal error.
  - Duplicate – indicates person ID already exists.
  - Error connecting to database – internal error.
  - Error during insert – internal error.
  - Missing Last Name
  - No record returned by nbapi.GetInsertIdFromPerson – internal error.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PERSONID as the identifier for the new person in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block
  - Duplicate person ID: 'XX'.
  - Missing LASTNAME

- The Role: full system setups does not exist.
- Duplicate person ID: '6'.
- ROLE is a mandatory field for AddPerson.
- PASSWORD is a mandatory field when AUTHTYPE value is either DB or LDAP.
- Missing ROLE.
- Missing AUTHTYPE Badging is not Enabled or Configured in the System. Cannot set Badge Print Request.
- Failed to save picture.
- Access level does not exist: XX
- ACTDATE and EXPDATE must follow YYYY-MM-DD hh:mm[:ss] or YYYY-MM-DD format

Example

Simple **AddPerson** call to add a person record for Frank Smith, with the person ID number 123, and two access levels:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="AddPerson" num="1">
        <PARAMS>
             <PERSONID>123</PERSONID>
             <FIRSTNAME>Frank</FIRSTNAME>
             <LASTNAME>Smith</LASTNAME>
             <USERNAME>frank</USERNAME>
             <PASSWORD>doug</PASSWORD>
             <ROLE>full system setup</ROLE>
             <AUTHTYPE>DB</AUTHTYPE>
             <ACCESSLEVELS>
                <ACCESSLEVEL>Access Level 1</ACCESSLEVEL>
                <ACCESSLEVEL>Access Level 2</ACCESSLEVEL>
             </ACCESSLEVELS>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **AddPerson** call adds the person record:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="AddPerson" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

More complex command using many of the **AddPerson** parameters:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="AddPerson" num="1">
        <PARAMS>
           <PERSONID>30001</PERSONID>
           <LASTNAME>Smith</LASTNAME>
           <FIRSTNAME>Frank</FIRSTNAME>
           <MIDDLENAME>V</MIDDLENAME>
           <EXPDATE>2024-01-01</EXPDATE>
           <ACTDATE>2016-09-11</ACTDATE>
```

```
          <ACCESSLEVELS>
             <ACCESSLEVEL>Access Level 1</ACCESSLEVEL>
             <ACCESSLEVEL>Access Level 2</ACCESSLEVEL>
          </ACCESSLEVELS>
          <UDF1>Job Title</UDF1>
          <UDF1>Manager</UDF1>
          <UDF2>Division</UDF2>
          <UDF3>Office Number</UDF3>
          <UDF4>Building Location</UDF4>
          <PIN>1111</PIN>
          <CONTACTPHONE>999-123-4567</CONTACTPHONE>
          <CONTACTEMAIL>sford@yahoo.com</CONTACTEMAIL>
          <CONTACTSMSEMAIL>sford@s2sys.com</CONTACTSMSEMAIL>
          <CONTACTLOCATION>Building 1</CONTACTLOCATION>
          <OTHERCONTACTNAME>John Brown</OTHERCONTACTNAME>
          <OTHERCONTACTPHONE1>999-123-4568</OTHERCONTACTPHONE1>
          <OTHERCONTACTPHONE2>999-123-4569</OTHERCONTACTPHONE2>
          <VEHICLES>
             <VEHICLE>
                <VEHICLECOLOR>Blue</VEHICLECOLOR>
                <VEHICLEMAKE>Honda</VEHICLEMAKE>
                <VEHICLEMODEL>Honda CRV</VEHICLEMODEL>
                <VEHICLESTATE>MA</VEHICLESTATE>
                <VEHICLELICNUM>123 PGA</VEHICLELICNUM>
                <VEHICLETAGNUM>TAG 3000</VEHICLETAGNUM>
             </VEHICLE>
             <VEHICLE>
                <VEHICLECOLOR>Red</VEHICLECOLOR>
                <VEHICLEMAKE>Buggatti</VEHICLEMAKE>
                <VEHICLEMODEL>XV32</VEHICLEMODEL>
                <VEHICLESTATE>MA</VEHICLESTATE>
                <VEHICLELICNUM>456 PGA</VEHICLELICNUM>
                <VEHICLETAGNUM>TAG 3001</VEHICLETAGNUM>
             </VEHICLE>
          </VEHICLES>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **AddPerson** call indicates that the person record was successfully added:

```
NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="AddPerson" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <PERSONID>30001</PERSONID>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

AddPortalGroup

The **AddPortalGroup** command adds a portal group.

Calling Parameters

- NAME: Portal group name of up to 64 characters.
- DESCRIPTION: Portal group description.
- UNLOCKTIMESPECGROUPKEY: Time spec group key for unlocking portals in this group.
- THREATLEVELGROUPKEY (optional): Threat level group key.
- PORTALKEYS: List of keys for portals to be included in the group.

Use **GetPortals** to retrieve a list of portal key values. Use **GetTimeSpecGroups** to retrieve a list of time spec groups values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PORTALGROUPKEY as the identifier for the new portal group in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - INVALID TIME SPEC (GROUP) KEY
  - PORTAL KEY <num> DOES NOT EXIST
  - Duplicate Portal Group
  - PORTAL GROUP NAME IS REQUIRED FIELD.
  - Portal Group Name exceeds is 64 characters
  - INVALID THREAT LEVEL GROUP KEY

Example

AddPortalGroup call to add a new portal group, "Bldg 10 Portal Group" with three portals. The command includes an UNLOCKTMESPECGROUPKEY, a THREATLEVELGROUPKEY, and a set of PORTALKEYS (which correspond to the three new portals).

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
   <COMMAND name="AddPortalGroup" num="1">
      <PARAMS>
           <NAME>Bldg 10 Portal Group</NAME>
           <DESCRIPTION>1</DESCRIPTION>
           <UNLOCKTIMESPECGROUPKEY>10</UNLOCKTIMESPECGROUPKEY>
           <THREATLEVELGROUPKEY>30<THREATLEVELGROUPKEY>
           <PORTALKEYS>
              <PORTALKEY>30</PORTALKEY>
              <PORTALKEY>32</PORTALKEY>
              <PORTALKEY>34</PORTALKEY>
           </PORTALKEYS>
```

```
      </PARAMS>
   </COMMAND>
</NETBOX-API>
```

Successful response to the **AddPortalGroup** call returns the new PORTALGROUPKEY:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddPortalGroup" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
           <PORTALGROUPKEY>42</PORTALGROUPKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddReaderGroup

The **AddReaderGroup** command adds a new reader group.

Calling Parameters

- NAME: Reader group name of up to 64 characters.
- DESCRIPTION: Reader group description.
- READERKEYS: List of keys for readers to be included in the group.

Use **GetReaders** to retrieve the list of READERKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the READERGROUPKEY as the identifier for the new reader group in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - READER with KEY <key> DOES NOT EXIST
  - Duplicate Reader Group
  - READER GROUP NAME IS REQUIRED FIELD
  - Reader Group Name exceeds 64 characters.

Example

AddReaderGroup adds a new reader group, named "Reader Group Building 10 First Floor." The command includes a list of READERKEYS that correspond to readers that will be included in the new reader group.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="AddReaderGroup" num="1">
        <PARAMS>
           <NAME>Reader Group Building 10 First Floor</NAME>
           <DESCRIPTION>Reader group first floor Bldg 10</DESCRIPTION>
           <READERKEYS>
              <READERKEY>2</READERKEY>
```

```
            <READERKEY>3</READERKEY>
            <READERKEY>5</READERKEY>
        </PARAMS>
</COMMAND>
</NETBOX-API>
```

Successful response to the **AddReaderGroup** call returns the new `READERGROUPKEY`:

```
NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddReaderGroup" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
           <READERGROUPKEY>41</READERGROUPKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddThreatLevel

The **AddThreatLevel** command adds a threat level. There are six default threat levels. Users can add two additional threat levels.

Calling Parameters

- `LEVELNAME`: Name of threat level.
- `SEQNUM` (optional): Display order for the threat level in the system's web interface.
- `COLOR` (optional): Specify White, Green, Blue, Yellow, Orange, or Red as desired.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Indicates the threat level has been successfully added.
- `FAIL`: Logs error message:
  - Duplicate – Indicates the threat level name already exists.

Example

AddThreatLevel call to request that a new threat level. "Intruder Alert", be added:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="AddThreatLevel" num="1">
        <PARAMS>
           <LEVELNAME>Intruder Alert</LEVELNAME>
           <SEQNUM>4</SEQNUM>
           <COLOR>Orange<COLOR>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **AddThreatLevel** call indicates that the threat level was added:

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddThreatLevel" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
```

```
</NETBOX>
```

AddThreatLevelGroup

The **AddThreatLevelGroup** command adds a threat level group.

Calling Parameters

- LEVELGROUPNAME: Name of the threat level group.
- LEVELNAMES (optional): Names of the threat levels to be added to this threat level group.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates the threat level group has been successfully added.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Name already exists.
  - Not Found – When a threat level is not found.

Example

This sample **AddThreatLevelGroup** call requests that a new threat level group, "Threat Level Intruder," is added. The new threat level group includes the threat levels: "Default," "Low," and "Guarded."

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="AddThreatLevelGroup" num="1">
        <PARAMS>
           <LEVELGROUPNAME>Threat Level Intruder</LEVELGROUPNAME>
           <LEVELNAMES>
              <LEVELNAME>Default</LEVELNAME>
              <LEVELNAME>Low</LEVELNAME>
              <LEVELNAME>Guarded</LEVELNAME>
           </LEVELNAMES>
        </PARAMS>
</COMMAND>
</NETBOX-API>
```

The successful response to the **AddThreatLevelGroup** call indicates that the threat level group was added.

```
NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddThreatLevelGroup" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

AddTimeSpec

The **AddTimeSpec** command adds a time spec.

Calling Parameters

- NAME: Name of the new time spec.
- DESCRIPTION: Description of the new time spec.
- STARTTIME: Start Time in HH:MM format.
- ENDTIME: End Time in HH:MM format.
- Days of the week to be included, 1 for TRUE and 0 for FALSE, for each of:

  MONDAY

  TUESDAY

  WEDNESDAY

  THURSDAY

  FRIDAY

  SATURDAY

  SUNDAY

- HOLIDAYGROUPS: Specify any or all of the holiday group numbers (1, 2, 3, 4, 5, 6, 7, 8 separated by commas).

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the TIMESPECKEY in the DETAILS element block as the identifier for the new time spec.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - BAD VALUE FOR WEEK DAY: MUST BE 1 OR 0
  - Time Spec name already exists: <Time spec name>
  - Time Spec name cannot be null
  - Time Spec name cannot be empty

Example

AddTimeSpec call includes the time spec definition for "Cleaning Crew."

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="AddTimeSpec" num="1">
       <PARAMS>
          <NAME>Cleaning Crew</NAME>
          <DESCRIPTION>Cleaning Crew hours</DESCRIPTION>
          <STARTTIME>16:00</STARTTIME>
          <ENDTIME>20:00</ENDTIME>
```

```
            <MONDAY>1</MONDAY>
            <TUESDAY>1</TUESDAY>
            <WEDNESDAY>1</WEDNESDAY>
            <THURSDAY>1</THURSDAY>
            <FRIDAY>1</FRIDAY>
            <SATURDAY>0</SATURDAY>
            <SUNDAY>0</SUNDAY>
            <HOLIDAYGROUPS>1</HOLIDAYGROUPS>
         </PARAMS>
      </COMMAND>
</NETBOX-API>
```

Successful response to the **AddTimeSpec** call returns a `TIMESPECKEY` with a value of 3:

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="AddTimeSpec" num="1">
        <DETAILS>
        <CODE>SUCCESS</CODE>
           <TIMESPECKEY>3</TIMESPECKEY>
     </DETAILS>
     </RESPONSE>
</NETBOX>
```

AddTimeSpecGroup

The **AddTimeSpecGroup** command adds a time spec group.

Calling Parameters

- `NAME`: Name of the new time spec group of up to 64 characters.
- `DESCRIPTION`: Description of the time spec group. Maximum size of the description is 255 characters.
- `TIMESPECKEYS`: Outer tag for list of timespec keys.
- `TIMESPECKEY`: Time spec key that can be added to new group that is being created.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the `TIMESPECGROUPKEY` in the `DETAILS` element block as the identifier for the new time spec group.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - DUPLICATE – indicates that the name already exists.
  - TIME SPEC GROUP NAME IS REQUIRED FIELD.
  - TIMESPECKEY with KEY <X> DOES NOT EXIST

Example

AddTimeSpecGroup call to add a new time spec group, "Maintenance Staff" containing three time specs:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
```

```
    <COMMAND name="AddTimeSpecGroup" num="1">
       <PARAMS>
          <NAME>Maintenance Staff</NAME>
          <DESCRIPTION> Maintenance Staff Flex Hours</DESCRIPTION>
          <TIMESPECKEYS>
             <TIMESPECKEY>15</TIMESPECKEY>
             <TIMESPECKEY>16</TIMESPECKEY>
             <TIMESPECKEY>17</TIMESPECKEY>
          </TIMESPECKEYS>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **AddTimeSpecGroup** call includes a `TIMESPECGROUPKEY` with a value of 7:

```
NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="AddTimeSpecGroup" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <TIMESPECGROUPKEY>7</TIMESPECGROUPKEY>
       /DETAILS>
    </RESPONSE>
</NETBOX>
```

AddVirtualCredentialRequest

The **AddVirtualCredentialRequest** command adds a mobile credential to a person record in the system.

Calling Parameters

- PERSONID: ID of the person associated with the credential to be added.
- CARDFORMAT: Name of the format to be used to decode the credential.

Note: PERSONID and CARDFORMAT are mandatory parameters.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the STATUS of the newly-added credential in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - CARDFORMAT NOT FOUND

    Cannot update Virtual Credential which is in an x state. Examples of states include Active, New, Delete Pending, Not Registered, Processing.
  - Cannot add or modify Virtual Credential when MSU is disabled for a person.

Example

AddVirtualCredentialRequest call to request assignment of mobile credential with a specific CARDFORMAT to person with PERSONID:

```
<NETBOX-API>
    <COMMAND name="AddVirtualCredentialRequest" num="1">
        <PARAMS>
            <PERSONID>123</PERSONID>
            <CARDFORMAT>LenelS2 48bit Corp 1000</CARDFORMAT>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to AddVirtualCredentialRequest call:

```
<NETBOX>
    <RESPONSE command="AddVirtualCredentialRequest" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <STATUS>Pending</STATUS>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

DeactivateOutput

The **DeactivateOutput** command requests the output specified by the OUTPUTKEY to deactivate.

Use **ActivateOutput** to activate an output.

Calling Parameters

- OUTPUTKEY: Key associated with the output to deactivate.
- Use **GetOutputs** to retrieve the list of OUTPUTKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the OUTPUTKEY associated with the output deactivation in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Invalid output key – the key provided does not match an output.
  - Missing OUTPUTKEY
  - Output not online or unreachable – the resource associated with the output is not available.
  - Output state not changed – the output was already in the specified state.
  - SQL function returned unexpected value – When the back end returns a result that does not match none of the above.

Example

DeactivateOutput call to deactivate the output associated with OUTPUTKEY 36:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="DeactivateOutput" num="1" dateformat="tzoffset">
        <PARAMS>
           <OUTPUTKEY>36</OUTPUTKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **DeactivateOutput** call indicates that the output associated with OUTPUTKEY 36 has been deactivated:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeactivateOutput" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <OUTPUTKEY>36</OUTPUTKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

DeleteAccessLevel

The **DeleteAccessLevel** command deletes an access level.

Calling Parameters

- ACCESSLEVELKEY: Key associated with the access level to be deleted.

  Use **GetAccessLevels** to retrieve the list of ACCESSLEVELKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates that the access level has been successfully deleted.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing ACCESSLEVELKEY
  - INVALID ACCESSLEVELKEY
  - NOT FOUND – indicates the access level key does not exist.

Example

DeleteAccessLevel call to deactivate the access level associated with an ACCESSLEVELKEY with a value of 12:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="DeleteAccessLevel" num="1">
       <PARAMS>
          <ACCESSLEVELKEY>12</ACCESSLEVELKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **DeleteAccessLevel** call indicates that the access level associated with the ACCESSLEVELKEY 12 has been deleted:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="DeleteAccessLevel" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <ACCESSLEVELKEY>12</ACCESSLEVELKEY>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

DeleteAccessLevelGroup

The **DeleteAccessLevelGroup** command deletes an access level group.

Calling Parameters

ACCESSLEVELGROUPKEY: Key associated with the access level group to be deleted.

Use **GetAccessLevelGroups** to retrieve the list of ACCESSLEVELGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates that the access level group has been successfully deleted.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Missing ACCESSLEVELGROUPKEY
  - INVALID ACCESSLEVELGROUPKEY

Example

DeleteAccessLevelGroup call to deactivate the access level associated with an ACCESSLEVELGROUPKEY with a value of 12:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="DeleteAccessLevelGroup" num="1">
        <PARAMS>
           <ACCESSLEVELGROUPKEY>12</ACCESSLEVELGROUPKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **DeleteAccessLevelGroup** call indicates that the access level group associated with the ACCESSLEVELGROUPKEY 12 has been deleted:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeleteAccessLevelGroup" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

DeleteHoliday

The **DeleteHoliday** command deletes an existing holiday.

Calling Parameters

- `HOLIDAYKEY`: Key associated with the holiday to be deleted.

  Use **GetHoliday** and **GetHolidays** to retrieve the list of `HOLIDAYKEY` values.

Response

- `SUCCESS`: Indicates that the holiday has been successfully deleted.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Missing HOLIDAYKEY
  - NOT FOUND
  - INVALID HOLIDAYKEY

Example

DeleteHoliday call to delete the holiday associated with a `HOLIDAYKEY` with a value of 5:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="DeleteHoliday" num="1">
       <PARAMS>
          <HOLIDAYKEY>5</HOLIDAYKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **DeleteHoliday** call indicates that the holiday associated with a `HOLIDAYKEY 5` has been deleted:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="DeleteHoliday" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <HOLIDAYKEY>5</HOLIDAYKEY>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

DeletePortalGroup

The **DeletePortalGroup** command deletes a portal group.

Calling Parameters

- PORTALGROUPKEY: Key associated with the portal group to be deleted.

  Use **GetPortalGroups** to retrieve the list of PORTALGROUPKEY values.

Response

- SUCCESS: Indicates that the portal group has been successfully deleted.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing PORTALGROUPKEY
  - INVALID PORTALGROUPKEY
  - NOT FOUND – indicates the key does not exist.

Example

DeletePortalGroup call to delete the portal group associated with a PORTALGROUPKEY with a value of 5:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="DeletePortalGroup" num="1">
        <PARAMS>
           <PORTALGROUPKEY>5</PORTALGROUPKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>

<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeletePortalGroup" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

DeleteReaderGroup

The **DeleteReaderGroup** command deletes a reader group.

Calling Parameters

- READERGROUPKEY: Key ASSOCIATED with the reader group to be deleted.

  Use **GetReaderGroups** to retrieve the list of READERGROUPKEY values.

Response

- SUCCESS: Indicates that the reader group has been successfully deleted.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing READERGROUPKEY
  - NOT FOUND– indicates that the reader group key does not exist.
  - INVALID READERGROUPKEY

Example

DeleteReaderGroup call to delete the reader group associated with a READERGROUPKEY with the value of 10:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="DeleteReaderGroup" num="1" dateformat="tzoffset">
        <PARAMS>
           <READERGROUPKEY>10</READERGROUPKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **DeleteReaderGroup** call indicates that the reader group associated with the READERGROUPKEY with a value of 10, has been deleted:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeleteReaderGroup" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <READERGROUPKEY>10</READERGROUPKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

DeleteTimeSpec

The **DeleteTimeSpec** command deletes a time spec.

Calling Parameters

- TIMESPECKEY: Key associated with the time spec to be deleted.

  Use **GetTimeSpecs** to retrieve the list of TIMESPECKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates that the time spec has been successfully deleted.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing TIMESPECKEY.
  - Invalid TIMESPECKEY.
  - Not Found – indicates that the time spec key does not exist.

Example

DeleteTimeSpec call to delete the time spec associated with a TIMESPECKEY with the value of 12:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="DeleteTimeSpec" num="1">
        <PARAMS>
           <TIMESPECKEY>12</TIMESPECKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **DeleteTimeSpec** call indicates that the time spec associated with the TIMESPECKEY with a value of 12, has been deleted:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeleteTimeSpec" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <TIMESPECKEY>12</TIMESPECKEY>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

DeleteTimeSpecGroup

The **DeleteTimeSpecGroup** command deletes a time spec group.

Calling Parameters

- TIMESPECGROUPKEY: Key associated with the time spec to be deleted.

  Use **GetTimeSpecGroups** to retrieve the list of TIMESPECGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates that the time spec group has been successfully deleted.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing TIMESPECGROUPKEY
  - INVALID TIMESPECGROUPKEY
  - NOT FOUND
  - Time Spec '<TimeSpec Name>' is in use.

Example

DeleteTimeSpecGroup call to delete the time spec group associated with a TIMESPECKEYGROUPKEY with the value of 12:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="DeleteTimeSpecGroup" num="1">
        <PARAMS>
           <TIMESPECGROUPKEY>12</TIMESPECGROUPKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **DeleteTimeSpecGroup** call indicates that the time spec group associated with the TIMESPECGROUPKEY with a value of 12, has been deleted:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeleteTimeSpecGroup" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

NBAPI version 2

This command returns SUCCESS or FAIL in the CODE element:

```
 <NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="DeleteTimeSpecGroup" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

DogOnNextExitPortal

The **DogOnNextExitPortal** command sets a portal specified by a PORTALKEY to the Dog On Next Exit state.

Use **UnlockPortal** to request that a portal be set to the Extended Unlock state.

Calling Parameters

- PORTALKEY: Key associated with the portal to be set to the Dog On Next Exit state.

  Use **GetPortalGroups** to retrieve the list of PORTALKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PORTALKEY associated with the portal to be set to the Dog On Next Exit state in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing PORTALKEY
  - Portal not online or unreachable – the resource associated with the portal is unavailable.
  - Portal state not changed – the portal was already in the specified state.
  - Command not supported for this device type.
  - SQL function returned unexpected value.

Example

DogOnNextExitPortal call to set to the Dog On Next Exit state the portal associated with a PORTALKEY with the value of 27:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
<COMMAND name="DogOnNextExitPortal" num="1">
<PARAMS>
<PORTALKEY>27</PORTALKEY>
</PARAMS>
</COMMAND>
</NETBOX-API>
```

Successful response to the **DogOnNextExitPortal** call indicates that the portal associated with a PORTALKEY with the value of 27 has been set to the Dog On Next Exit state:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="DogOnNextExitPortal" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PORTALKEY>27</PORTALKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetAccessHistory

The **GetAccessHistory** command retrieves a history of accesses of data log records from all users or for a specific access card at a specific point in time, new accesses that have occurred, and access history for a user-defined time span.

By default, a system maintains a maximum of 100,000 Activity Log records in the active database for reporting purposes. You can increase this maximum by entering a new number in the Maximum number of Activity Log entries maintained in active database text field. This field is located in the Activity Log section on the *Events and Activity* tab of the *Network Controller* page.

The **GetAccessHistory** command is a supplement to the **GetCardAccessDetails** command.

Calling Parameters

- `STARTLOGID` (optional): Used to start retrieval with a specific log ID. This command is used in conjunction with `NEXTLOGID` returned from a prior call.

  Use **GetCardAccessDetails** to retrieve the log IDs.

- `AFTERLOGID` (optional): Used to start retrieval after a specific log ID that was previously returned. Implies an `ASCENDING` order of returned log IDs.
- `ORDER` (optional): Specifies the order of returned log IDs as either `ASCENDING` or `DESCENDING`. The default is `DESCENDING` unless `AFTERLOGID` is supplied.
- `MAXRECORDS` (optional): Maximum number of `ACCESSENTRY` records returned in one **GetAccessHistory** call. If not supplied, all access records matching the search criteria are returned.
- `ENCODEDNUM` (optional): Used to identify a specific card, together with the `CARDFORMAT`.

  Note: Specify either `ENDCODEDNUM` or `HOTSTAMP` with `CARDFORMAT` to identify a specific card.

- `HOTSTAMP` (optional): Used to identify a specific card, together with the `CARDFORMAT`.
- `CARDFORMAT` (optional): Name of the card format.
- `OLDESTDTTM` (optional): Specifies the oldest access date that the command will retrieve. See *Date Formats*.
- `NEWESTDTTM` (optional): Specifies the most recent access date that the command will retrieve. See *Date Formats*.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS:` Returns the following parameters in the `DETAILS` element block:
  - `LOGID:` Number which identifies the data log record.

- PERSONID: External Person ID associated with the person who owns the specified credential. This is the field in the person record labeled "ID #."
- READER: Name of the card reader.
- DTTM: System date and time associated with the data log. See *Date Formats*.
- NODEDTTM: Node date and time associated with the data log. See *Date Formats*.
- TYPE: Reason type which specifies a valid or invalid access. And invalid access also returns a Reason code. See
- *Type* Element for details on the Reason types and Reason codes.
- NEXTLOGID: Unique identifier for the next log ID that can be retrieved from the system. This number is used to specify the next log ID in STARTLOGID in the next **GetAccessHistory** command.
- PORTALNAME: NAME OF THE PORTAL ASSOCIATED WITH THE DATA LOG.
- PORTALKEY: Unique identifier for the portal.
- READERKEY: Unique identifier for the reader.

- FAIL: Returns a Reason TYPE and CODE .

Example

This example of a **GetAccessHistory** command returns the most recent access history records in reverse time order (most recent first) up to the internal maximum configured on your system.

The initial call:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessHistory" num="1">
    </COMMAND>
</NETBOX-API>
```

The response from the initial call with the TYPE code value of 1 indicating a valid access:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessHistory" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <ACCESSES>
             <ACCESS>
                <LOGID>56957</LOGID>
                <PERSONID>37_KD</PERSONID>
                <READER>Stair 1 3rd Floor Emp. Entrance</READER>
                <DTTM>2016-10-19 11:59:11</DTTM>
                <NODEDTTM>2016-10-19 11:59:10</NODEDTTM>

                <TYPE>1</TYPE>
                <REASON></REASON>
                <READERKEY>19</READERKEY>
                <PORTALKEY>5</PORTALKEY>
             </ACCESS>
             <ACCESS>
                <LOGID>56951</LOGID>
```

```
                <PERSONID>90_JP</PERSONID>
                <READER>Lobby Door</READER>
                <DTTM>2016-10-19 11:37:16</DTTM>
                <NODEDTTM>2016-10-19 11:37:15</NODEDTTM>
                   <TYPE>1</TYPE>
                <REASON></REASON>
                <READERKEY>16</READERKEY>
                <PORTALKEY>4</PORTALKEY>
             </ACCESS>
             <ACCESS>
                <LOGID>56948</LOGID>
                <PERSONID>76_RN</PERSONID>
                <READER>Stair 1 3rd Floor Emp. Entrance</READER>
                <DTTM>2016-10-19 11:37:06</DTTM>
                <NODEDTTM>2016-10-19 11:37:05</NODEDTTM>
                <TYPE>1</TYPE>
                <REASON></REASON>
                <READERKEY>19</READERKEY>
                <PORTALKEY>5</PORTALKEY>
             </ACCESS>
                   .
                   .
                   .
          </ACCESSES>
          <NEXTLOGID>53587</NEXTLOGID>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

Subsequent call using the `NEXTLOGID` value will retrieve the next older chunk of log records:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessHistory" num="1" dateformat="tzoffset">
       PARAMS>
          <STARTLOGID>53587</STARTLOGID>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Another call later using `AFTERLOGID` will retrieve any records which have been created since the first call:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessHistory" num="1" dateformat="tzoffset">
      <PARAMS>
         <AFTERLOGID>56957</AFTERLOGID>
      </PARAMS>
    </COMMAND>
</NETBOX-API>
```

These would be returned in ascending order, going forwards, for example:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command=" GetAccessHistory" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <ACCESSES>
             <ACCESS>
                <LOGID>56958</LOGID>
```

```
                <PERSONID>uid4</PERSONID>
                <READER>reader 2</READER>
                <DTTM>2006-06-23 10:31:06 -0400</DTTM>
                <TYPE>1</TYPE>
                <READERKEY>5</READERKEY>
                <PORTALKEY>30</PORTALKEY>
             </ACCESS>
          </ACCESSES>
          <NEXTLOGID>-1</NEXTLOGID>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

This indicates that only 1 new entry was found. Suppose that another call is made.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessHistory" num="1" dateformat="tzoffset">
      <PARAMS>
          <AFTERLOGID>56958</AFTERLOGID>
      </PARAMS>
    </COMMAND>
</NETBOX-API>
```

If there are no more, it will return "`NOT FOUND`", for example:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command=" GetAccessHistory" num="1">
       <CODE>NOT FOUND</CODE>
    </RESPONSE>
</NETBOX>
```

GetAccessLevel

The **GetAccessLevel** command retrieves information about an access level.

Calling Parameters

- ACCESSLEVELKEY: The access level key.

  Use **GetAccessLevels** to retrieve the list of ACCESSLEVELKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - ACCESSLEVELNAME: Name of the access level.
  - ACCESSLEVELDESCRIPTION: Access level description.
  - READERGROUPKEY: Key corresponding to the reader group assigned to this access level.
  - TIMESPECGROUPKEY: Key corresponding to the time spec group assigned to this access level.
  - THREATLEVELGROUPKEY: Key corresponding to the threat level group assigned to this access level.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - Missing ACCESSLEVELKEY.
  - Invalid ACCESSLEVELKEY
  - Not Found – indicates the access level key does not exist.

Example

GetAccessLevel command to retrieve the definition of the access level named, "24/7/365 Access":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevel" num="1">
        <PARAMS>
            <ACCESSLEVELKEY>1</ACCESSLEVELKEY>
          </PARAMS>
       </COMMAND>
</NETBOX-API>
```

Successful response to the **GetAccessLevel** call provides the definition of the requested access level:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevel" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELNAME>24/7/365 Access</ACCESSLEVELNAME>
            <ACCESSLEVELDESCRIPTION></ACCESSLEVELDESCRIPTION>
            <READERGROUPKEY>21</READERGROUPKEY>
            <TIMESPECGROUPKEY>1</TIMESPECGROUPKEY>
            <THREATLEVELGROUPKEY></THREATLEVELGROUPKEY>
```

```
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetAccessLevelGroup

The **GetAccessLevelGroup** command retrieves information about an access level group.

Calling Parameters

- ACCESSLEVELGROUPKEY: The access level group key.

  Use **GetAccessLevelGroups** to retrieve the list of ACCESSLEVELGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - ACCESSLEVELGROUPKEY: Access level group key specified.
  - NAME: Name of the access level group.
  - DESCRIPTION: Access level group description.
    - Either of PARTITIONKEY or SYSTEMGROUP:
    - PARTITIONKEY: The partition the access level group belongs to.
    - SYSTEMGROUP: Indicates multi-partition access level group.
  - ACCESS LEVELS:

    List of ACCESSLEVEL definitions that include KEY and NAME and, conditionally, PARTITIONKEY when the group is a multi-partition access level group.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - Invalid ACCESSLEVELGROUPKEY
  - Access Level Group key not found
  - Failure obtaining AccessLevelGroup partition id
  - Missing ACCESSLEVELGROUPKEY

Example

GetAccessLevelGroup command to retrieve the definition of the access level group named, "Exterior Doors" (where the KEY of 21 was obtained from **GetAccessLevelGroups**):

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevelGroup" num="1">
        <PARAMS>
            <ACCESSLEVELGROUPKEY>21</ACCESSLEVELGROUPKEY>
        </PARAMS>
    </COMMAND>
```

```
</NETBOX-API>
```

Successful response to the **GetAccessLevelGroup** call provides the definition of the requested access level group:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevel" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELGROUPKEY>21</ACCESSLEVELGROUPKEY>
            <NAME>Exterior Doors</NAME>
            <DESCRIPTION></DESCRIPTION>
            <PARTITIONKEY>2</PARTITIONKEY>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <KEY>15</KEY>
                    <NAME>Front Doors</NAME>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>16</KEY>
                    <NAME>Back Doors</NAME>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

Successful response to a **GetAccessLevelGroup** call that provides a multi-partition access level group definition:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevel" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELGROUPKEY>35</ACCESSLEVELGROUPKEY>
            <NAME>Lobby Doors</NAME>
            <DESCRIPTION></DESCRIPTION>
            <SYSTEMGROUP>1</SYSTEMGROUP>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <KEY>85</KEY>
                    <NAME>Framingham Lobby</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>103</KEY>
                    <NAME>Boston Lobby Entrance</NAME>
                    <PARTITIONKEY>5</PARTITIONKEY>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetAccessLevelGroups

The **GetAccessLevelGroups** command returns a list of access level groups with their corresponding group key, description and name, and access level member keys and names. Optionally, the command can be used to include a multi-partition list which would also include the partition of each access level.

Full system setup user role is required to retrieve access levels across partitions.

Partition setup or user role mapping is required to retrieve access levels for specific partitions.

Calling Parameters

- (Optional) `PARTITIONKEY`: Returns access levels for all partitions. Only a value of 0 is allowed, all other values will result in command failure.

  The `PARTITIONKEY` parameter requires User Login Authentication and the account that the API client uses to log in must have Full system setup privilege.

  When the `PARTITIONKEY` value is 0, the response will contain a list of access levels with `KEY`, `NAME`, and `PARTITIONKEY` elements for each access level.

  When the `PARTITIONKEY` parameter is omitted, the response will contain a list of access level names OR access level keys.

- startfromKEY: Specifies the starting `ACCESSLEVELGROUP` key.

  `STARTFROMKEY` is used in conjunction with `NEXTKEY` to retrieve the next set of access level group keys or access levels ordered by key. If `NEXTKEY` is returned with a value greater than 0, a `STARTFROMKEY` parameter value can be passed in the next **GetAccessLevelGroups** call to return the next set of access level keys.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the following parameters in the `DETAILS` element block:
  - `ACCESSLEVELGROUPS`: Element block that contains the `ACCESSLEVELGROUP` elements.
  - `ACCESSLEVEL`: If `PARTITIONKEY` is not supplied in the request, this tag contains the access level group `NAME` and `KEY`. If `PARTITIONKEY` is supplied, this tag includes the `PARTITIONKEY` of the access level.
  - `NEXTKEY` is returned with a value of -1, if there are no more access levels to return.
    If `STARTFROMKEY` is omitted, all access level definitions are returned.
  - `NEXTNAME` is returned with no value, if there are no more access level names to return.
    If `STARTFROMNAME` is omitted, all access level definitions are returned.

- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - If specified, `PARTITIONKEY` value must be 0
  - Invalid `STARTFROMKEY`

Example

This sample **GetAccessLevelGroups** command retrieves all access level groups, starting with the first group, by specifying a `STARTFROMKEY` value of 0.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevelGroups" num="1">
       <PARAMS>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevelGroups** call provides a list of access level group keys, names.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevelGroups" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELGROUPS>
                <ACCESSLEVELGROUP>
                    <KEY>25</KEY>
                    <NAME>24/7/365 Access</NAME>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>28</KEY>
                    <NAME>All Doors / All Times</NAME>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>29</KEY>
                    <NAME>AMR Access Level</NAME>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>35</KEY>
                    <NAME>Chicago Access Level</NAME>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>45</KEY>
                    <NAME>CL Arm DMP</NAME>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>49</KEY>
                    <NAME>CL Arm DMP2</NAME>
                </ACCESSLEVELGROUP>
            </ACCESSLEVELGROUPS>
            <NEXTKEY>-1</STARTFROMKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns -1 for `STARTFROMKEY`, indicates that there are no more access level groups to return.

This sample **GetAccessLevelGroups** command retrieves all access level groups for all partitions by specifying a `PARTITIONKEY` of 0.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevelGroups" num="1">
       <PARAMS>
          <PARTITIONKEY>0</PARTITIONKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevelGroups** call with `PARTITIONKEY` parameter provides a list of access level group names, keys and partition keys.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevelGroups" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELGROUPS>
                <ACCESSLEVELGROUP>
                    <KEY>25</KEY>
                    <NAME>24/7/365 Access</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>28</KEY>
                    <NAME>All Doors / All Times</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>29</KEY>
                    <NAME>AMR Access Level</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>35</KEY>
                    <NAME>Chicago Access Level</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>45</KEY>
                    <NAME>CL Arm DMP</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVELGROUP>
                <ACCESSLEVELGROUP>
                    <KEY>49</KEY>
                    <NAME>CL Arm DMP2</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVELGROUP>
            </ACCESSLEVELGROUPS>
            <NEXTKEY>-1</STARTFROMKEY>
```

```
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetAccessLevelNames

The **GetAccessLevelNames** command returns a list of names of access that can be assigned to people.

Note: This function should be used instead of **GetAccessLevels** to obtain the full set of valid access levels for assignment.

The full system setup user role is required to retrieve access level names across partitions.

Partition setup or user role mapping is required to retrieve access level names for specific partitions.

Calling Parameters

- PARTITIONKEY: (optional): Returns access levels for all partitions. Only a value of 0 is allowed, all other values will result in command failure.

  The PARTITIONKEY parameter requires the account that the API client uses to log in must have Full system setup privilege. APIERROR value of 5 is returned otherwise. User Login Authentication is not required.

  When the PARTITIONKEY value is 0, the response will contain a list of access levels with NAME and PARTITIONKEY elements for each access level.

  When the PARTITIONKEY parameter is omitted, the response will contain a list of access level names OR access level keys.

- STARTFROMNAME: Specifies the starting ACCESSLEVEL name.

  STARTFROMNAME is used in conjunction with NEXTNAME to retrieve the next set of access level names or access levels ordered by key. If NEXTNAME is returned with a value of 1 or greater, a STARTFROMNAME parameter value can be passed in the next **GetAccessLevelNames** call to return the next set of access level names.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - ACCESSLEVELS: Element block that contains the ACCESSLEVEL elements.
  - ACCESSLEVEL: If partition information is not supplied, this tag contains the access level NAME. When partition information is requested, this tag contains the NAME and PARTITIONKEY.
  - NEXTNAME is returned with no value, if there are no more access level names to return. If STARTFROMNAME is omitted, all access level definitions are returned.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - If specified, PARTITIONKEY value must be 0

- NOT FOUND

Example

This sample **GetAccessLevelNames** command retrieves the access level names.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetAccessLevelNames" num="1">
        <PARAMS>
           <STARTFROMNAME></STARTFROMNAME>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevelNames** call provides a list of access level names.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevelNames" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>24/7/365 Access</ACCESSLEVEL>
                <ACCESSLEVEL>All Doors / All Times</ACCESSLEVEL>
                <ACCESSLEVEL>AMR Access Level</ACCESSLEVEL>
                <ACCESSLEVEL>Chicago Access Level</ACCESSLEVEL>
                <ACCESSLEVEL>CL Arm DMP</ACCESSLEVEL>
                <ACCESSLEVEL>Common Areas</ACCESSLEVEL>
                <ACCESSLEVEL>Disarm DMP</ACCESSLEVEL>
                <ACCESSLEVEL>EMPLOYEE ACCESS</ACCESSLEVEL>
                <ACCESSLEVEL>Master Night Access</ACCESSLEVEL>
                <ACCESSLEVEL>Night CleanUP Crew Access</ACCESSLEVEL>
                <ACCESSLEVEL>Office Personnel</ACCESSLEVEL>
                <ACCESSLEVEL>Perimeter Access Level</ACCESSLEVEL>
                <ACCESSLEVEL>Patriot M-S 6AM-7PM</ACCESSLEVEL>
                <ACCESSLEVEL>Sample access level 1</ACCESSLEVEL>
                <ACCESSLEVEL>Tenants ABC Shared Access</ACCESSLEVEL>
                <ACCESSLEVEL>Visitor limited time</ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTNAME></NEXTNAME>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns no value for `NEXTNAME`, indicates that there are no more access levels to return.

This sample **GetAccessLevelNames** command retrieves all access levels for all partitions by specifying a `PARTITIONKEY` of 0.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetAccessLevelNames" num="1">
        <PARAMS>
           <PARTITIONKEY>0</PARTITIONKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevelNames** call provides a list of access levels sorted by name.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
         <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <NAME>AL_1_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL1_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL_2_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL2_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL_3_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL3_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL_4_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL4_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <NAME>AL_5_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTNAME></NEXTNAME>
        </DETAILS>
    </RESPONSE>
<NETBOX>
```

GetAccessLevels

The **GetAccessLevels** command returns a list of access levels or access level keys. Optionally, the command can be used to include a multi-partition list which would also include the partition of each access level.

Full system setup user role is required to retrieve access levels across partitions.

Partition setup or user role mapping is required to retrieve access levels for specific partitions.

Calling Parameters

- PARTITIONKEY: RETURNS ACCESS LEVELS FOR ALL PARTITIONS. ONLY A VALUE OF 0 IS ALLOWED, ALL OTHER VALUES WILL RESULT IN COMMAND FAILURE.

  THE PARTITIONKEY PARAMETER REQUIRES THE ACCOUNT THAT THE API CLIENT USES TO LOG IN MUST HAVE FULL SYSTEM SETUP PRIVILEGE. USER LOGIN AUTHENTICATION IS NOT REQUIRED.

  WHEN THE PARTITIONKEY VALUE IS 0, THE RESPONSE WILL CONTAIN A LIST OF ACCESS LEVELS WITH KEY, NAME, AND PARTITIONKEY ELEMENTS FOR EACH ACCESS LEVEL.

  WHEN THE PARTITIONKEY PARAMETER IS OMITTED, THE RESPONSE WILL CONTAIN A LIST OF ACCESS LEVEL NAMES OR ACCESS LEVEL KEYS.

- `STARTFROMKEY`: Specifies the starting `ACCESSLEVEL` key.

  `STARTFROMKEY` is used in conjunction with `NEXTKEY` to retrieve the next set of access level keys or access levels ordered by key. If `NEXTKEY` is returned with a value greater than 0, a `STARTFROMKEY` parameter value can be passed in the next **GetAccessLevels** call to return the next set of access level keys.

- `STARTFROMNAME`: Specifies the starting `ACCESSLEVEL` name.

  `STARTFROMNAME` is used in conjunction with `NEXTNAME` to retrieve the next set of access level names or access levels ordered by key. If `NEXTNAME` is returned with a value of 1 or greater, a `STARTFROMNAME` parameter value can be passed in the next **GetAccessLevels** call to return the next set of access level names.

- `WANTKEY`: (NetBox Only): If `TRUE` is supplied, access levels keys or sets of KEY, NAME, and PARTITION KEY elements ordered by access level keys are returned. If FALSE or WANTKEY is omitted access levels names or sets of KEY, NAME, and PARTITION KEY elements ordered by access level names are returned.
  - For a valid request ordered by access level key, the WANTKEY element value must be TRUE if used with the STARTFROMKEY element.
  - For a valid request ordered by name, the WANTKEY element value must be FALSE if used with the STARTFROMNAME element.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the following parameters in the `DETAILS` element block:
  - `ACCESSLEVELS`: Element block that contains the `ACCESSLEVEL` elements.
  - `ACCESSLEVEL`: If partition information is not supplied, this tag contains the access level name or key based on the calling convention. When partition information is requested, this tag contains the sub-tags `NAM`, `KEY`, and `PARTITIONKEY`.

  - NEXTKEY is returned with a value of -1, if there are no more access levels to return. If STARTFROMKEY is omitted, all access level definitions are returned.
  - NEXTNAME is returned with no value, if there are no more access level names to return. If STARTFROMNAME is omitted, all access level definitions are returned.

- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - GetAccessLevels PartitionKey value must be numeric 0

Example

This sample **GetAccessLevels** command retrieves the access level names using a STARTFROMKEY value
of 0.

```
<NETBOX-API sessionid="898755285">
    <COMMAND name='GetAccessLevels' num="1">
       <PARAMS>
          <WANTKEY>TRUE</WANTKEY>
          <STARTFROMKEY>0</STARTFROMKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevels** call provides a list of access level names.

```
<NETBOX sessionid="898755285">
    <RESPONSE command="GetAccessLevels" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>1</ACCESSLEVEL>
                <ACCESSLEVEL>2</ACCESSLEVEL>
                <ACCESSLEVEL>3</ACCESSLEVEL>
                <ACCESSLEVEL>4</ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns no value for NEXTNAME, indicates that there are no more access levels to return.

This sample **GetAccessLevels** command retrieves the access level keys using a WANTKEY of "TRUE."

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevels" num="1">
       <PARAMS>
          <WANTKEY>TRUE</WANTKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevels** call provides a list of access level keys.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevels" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>1</ACCESSLEVEL>
                <ACCESSLEVEL>2</ACCESSLEVEL>
                <ACCESSLEVEL>3</ACCESSLEVEL>
                <ACCESSLEVEL>4</ACCESSLEVEL>
                <ACCESSLEVEL>5</ACCESSLEVEL>
                <ACCESSLEVEL>6</ACCESSLEVEL>
                <ACCESSLEVEL>7</ACCESSLEVEL>
                <ACCESSLEVEL>8</ACCESSLEVEL>
                <ACCESSLEVEL>9</ACCESSLEVEL>
                <ACCESSLEVEL>10</ACCESSLEVEL>
                <ACCESSLEVEL>11</ACCESSLEVEL>
                <ACCESSLEVEL>12</ACCESSLEVEL>
                <ACCESSLEVEL>13</ACCESSLEVEL>
                <ACCESSLEVEL>14</ACCESSLEVEL>
                <ACCESSLEVEL>15</ACCESSLEVEL>
                <ACCESSLEVEL>19</ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns a `NEXTKEY` value of -1, which indicates that there are no more access levels to return.

This sample **GetAccessLevels** command retrieves all access levels for all partitions by specifying a `PARTITIONKEY` of 0 and a `WANTKEY` of `FALSE`.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevels" num="1">
       <PARAMS>
          <PARTITIONKEY>0</PARTITIONKEY>
          <WANTKEY>FALSE</WANTKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevels** call provides a list of access levels sorted by access level name.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
         <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <KEY>1</KEY>
                    <NAME>AL_1_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>700</KEY>
                    <NAME>AL1_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
```

```
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>2</KEY>
                    <NAME>AL_2_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>701</KEY>
                    <NAME>AL2_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>3</KEY>
                    <NAME>AL_3_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>702</KEY>
                    <NAME>AL3_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>600</KEY>
                    <NAME>AL_4_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>703</KEY>
                    <NAME>AL4_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>601</KEY>
                    <NAME>AL_5_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>704</KEY>
                    <NAME>AL5_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTNAME></NEXTNAME>
        </DETAILS>
```

This sample **GetAccessLevels** command retrieves all access levels for all partitions by specifying a `PARTITIONKEY` of 0 and a `WANTKEY` of `TRUE`.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevels" num="1">
       <PARAMS>
          <PARTITIONKEY>0</PARTITIONKEY>
          <WANTKEY>FALSE</WANTKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the **GetAccessLevels** call provides a list of access level for all partitions sorted by access level key.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
<DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <KEY>1</KEY>
                    <NAME>AL_1_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>2</KEY>
                    <NAME>AL_2_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>3</KEY>
                    <NAME>AL_3_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>600</KEY>
                    <NAME>AL_4_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>601</KEY>
                    <NAME>AL_5_P1</NAME>
                    <PARTITIONKEY>1</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>700</KEY>
                    <NAME>AL1_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>701</KEY>
                    <NAME>AL2_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>702</KEY>
                    <NAME>AL3_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>703</KEY>
                    <NAME>AL4_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
                <ACCESSLEVEL>
                    <KEY>704</KEY>
                    <NAME>AL5_P2</NAME>
                    <PARTITIONKEY>2</PARTITIONKEY>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
```

```
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
```

GetAPIVersion

The **GetAPIVersion** command retrieves the current version of the API from the server.

Calling Parameters

The **GetAPIVersion** command has no calling parameters.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the following parameters in the `DETAILS` element block:
  - `APIVERSION`: Alpha-numeric string with current version of the API software.
- `FAIL`: Returns no error message in the `ERRMSG` element in the `DETAILS` element block.

Example

GetAPIVersion call to return the current API software version:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetAPIVersion" num="1">
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetAPIVersion** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetAPIVersion" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <APIVERSION>4.7</APIVERSION>
           </DETAILS>
     </RESPONSE>
</NETBOX>
```

GetCardAccessDetails

The **GetCardAccessDetails** command retrieves recent card access events and related information associated with a given credential as defined by that credential's ENCODEDNUM and CARDFORMAT parameters.

Calling Parameters

- ENCODEDNUM: Used to identify a specific card, together with the CARDFORMAT.
- CARDFORMAT: Name of the card format.
- MAXRECORDS (optional): Maximum number of ACCESSENTRY records returned in one call. If not supplied, all access records matching the search criteria are returned. To use the function to retrieve only the person owning the credential, specify zero (0).
- OLDESTDTTM (optional): Specifies the oldest access date that the command will retrieve. See *Date Formats*.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in DETAILS element block:
  - PERSONID: External Person ID associated with the person who owns the specified credential. This is the field in the person record labeled "ID #."
  - DISABLED: Returns 1 if the credential is currently marked disabled or 0 if it is not.
  - EXPDATE: Expiration date of the credential requested in this command.
  - ACCESSES: The ACCESSES element block contains access log entry information in separate ACCESS element blocks.

    Each ACCESS element block contains the following command responses:

    - DTTM: System date and time associated with the data log. See *Date Formats*.
    - NODEDTTM: Node data and time associated with the data log. See *Date Formats*.
    - TYPE: Reason type which specifies a valid or invalid access. An invalid access also returns a Reason code. See
    - *Type* Element for details.
    - PORTALNAME: Name of the portal associated with the data log.
    - REASON: Reason code for command failure. See
    - *Type* Element for details.
    - PORTALKEY: Unique identifier for the portal.
    - READERKEY: Unique identifier for the reader.
    - LOGID: Number which identifies the data log record.

  - NEXTLOGID: Unique identifier for the next log ID that can be retrieved. This number is used to specify the next log ID in STARTLOGID in the next **GetCardAccessDetails** command.

- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - GETCARDACCESSDETAILS COMMAND REQUIRES CARD_FORMAT AND ENCODEDNUM ELEMENTS.
  - MISSING REQUIRED PARAMETER. MUST SUPPLY EITHER ENCODED NUMBER OR HOTSTAMP WITH CARD FORMAT, OR CREDENTIAL ID.

Example

GetCardAccessDetails call to request information on the last four card access events for a person record with ENCODEDNUM of 1202 and a CARDFORMAT called "W26-FC16":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">>
    <COMMAND name="GetCardAccessDetails" num="1">
       <PARAMS>
          <ENCODEDNUM>1202</ENCODEDNUM>
          <CARDFORMAT>W26-FC16</CARDFORMAT>
          <MAXRECORDS>4</MAXRECORDS>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response from the **GetCardAccessDetails** call returns the requested information for the last four card access events:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">>
    <RESPONSE command="GetCardAccessDetails" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <PERSONID>79_RN</PERSONID>
          <DISABLED>0</DISABLED>
          <EXPDATE></EXPDATE>
          <ACCESSES>
             <ACCESS>
                <DTTM>2016-10-14 08:16:37</DTTM>
                <NODEDTTM>2016-10-14 08:16:35</NODEDTTM>
                <TYPE>1</TYPE>
                <PORTALNAME>Stair 1 3rd Floor Entrance</PORTALNAME>
                <REASON></REASON>
                <PORTALKEY>5</PORTALKEY>
                <READERKEY>19</READERKEY>
                <LOGID>55789</LOGID>
             </ACCESS>
             <ACCESS>
                <DTTM>2016-10-13 10:17:08</DTTM>
                <NODEDTTM>2016-10-13 10:17:07</NODEDTTM>
                <TYPE>1</TYPE>
                <PORTALNAME>Stair 1 3rd Floor Entrance</PORTALNAME>
                <REASON></REASON>
                <PORTALKEY>5</PORTALKEY>
                <READERKEY>19</READERKEY>
                <LOGID>55489</LOGID>
```

```
            </ACCESS>
            <ACCESS>
               <DTTM>2016-10-13 07:14:34</DTTM>
               <NODEDTTM>2016-10-13 07:14:32</NODEDTTM>
               <TYPE>1</TYPE>
               <PORTALNAME>Stair 1 3rd Floor Entrance</PORTALNAME>
               <REASON></REASON>
               <PORTALKEY>5</PORTALKEY>
               <READERKEY>19</READERKEY>
               <LOGID>55355</LOGID>
            </ACCESS>
            <ACCESS>
               <DTTM>2016-10-11 10:39:46</DTTM>
               <NODEDTTM>2016-10-11 10:39:45</NODEDTTM>
               <TYPE>1</TYPE>
               <PORTALNAME>Stair 1 3rd Floor Entrance</PORTALNAME>
               <REASON></REASON>
               <PORTALKEY>5</PORTALKEY>
               <READERKEY>19</READERKEY>
               <LOGID>54811</LOGID>
            </ACCESS>
         </ACCESSES>
         <DTTM></DTTM>
         <PORTALNAME> </PORTALNAME>
         <PORTALKEY></PORTALKEY>
         <READERKEY></READERKEY>
         <NEXTLOGID>54708</NEXTLOGID>
      </DETAILS>
   </RESPONSE>
</NETBOX>
```

GetCardFormats

The **GetCardFormats** command retrieves a list of the names of the defined card formats.

Calling Parameters

This command has no calling parameters.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns a list of names of card formats in the `CARDFORMATS` element block.
- `FAIL`: Returns no error message in the `ERRMSG` element in the `DETAILS` element block,

Example

GetCardFormats call to request the list of names of the card formats:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetCardFormats" num="1"></COMMAND>
</NETBOX-API>
```

Successful response to the **GetCardFormats** call returns a list of card formats:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetCardFormats" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <CARDFORMATS>
             <CARDFORMAT>26 bit Wiegand</CARDFORMAT>
             <CARDFORMAT>Corporate 1000 35 bit</CARDFORMAT>
             <CARDFORMAT>Casi Rusco 40 bit</CARDFORMAT>
             <CARDFORMAT>FIPS 201 75-bit</CARDFORMAT>
             <CARDFORMAT>Software House 37 bit Wiegand</CARDFORMAT>
             <CARDFORMAT>Lenel 36 bit</CARDFORMAT>
             <CARDFORMAT>Honeywell 40 bit</CARDFORMAT>
             <CARDFORMAT>36 Bit</CARDFORMAT>
             <CARDFORMAT>Remote Lockset PIN Only</CARDFORMAT>
             <CARDFORMAT>STRAC 128 bit</CARDFORMAT>
             <CARDFORMAT>FIPS 201 128-bit</CARDFORMAT>
             <CARDFORMAT>Corporate 1000 48 bit</CARDFORMAT>
          </CARDFORMATS>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetElevators

The **GetElevators** command returns a list of elevators.

Calling Parameters

- STARTFROMKEY (optional): The STARTROMKEY specifies the starting floor KEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of portals. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetElevators** call to return the next set of elevators.

  NEXTKEY is returned with a value of -1, if there are no more portals to return. If STARTFROMKEY is omitted, all portal definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - ELEVATORS: Element block that contains an ELEVATOR element block for each elevator.
  - ELEVATOR: Element block that contains a KEY, NAME and SEQUENCE element block for each elevator.
  - KEY: Elevator key.
  - NAME: Name of the elevator.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Invalid STARTFROMKEY

Example

```
GetElevators call to retrieve the elevators using a STARTFROMKEY value of
    0:
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetElevators" num="1">
       <PARAMS>
          <STARTFROMKEY>0</STARTFROMKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetElevators** call that provides a list of elevators:

```
<NETBOX sessionid="1949792770">
    <RESPONSE command="GetElevators" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <ELEVATORS>
                <ELEVATOR>
```

```
                    <KEY>1</KEY>
                    <NAME>Penthouse</NAME>
                 </ELEVATOR>
              </ELEVATORS>
              <NEXTKEY>-1</NEXTKEY>
           </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns a `NEXTKEY` value of -1, which indicates there are no more elevators to return.

GetEventHistory

The **GetEventHistory** command requests a list of activity log events.

By default, the system maintains a maximum of 100,000 Activity Log records in the active database for reporting purposes.

You can increase this maximum by entering a new number in the *Maximum number of Activity Log entries maintained in active database* field. This field is located in the *Activity Log* section of the *Events and Activity* tab of the *Network Controller* page. This api returns a maximum of 1000 records between the specified dates.

Calling Parameters

- `EVENTNAME` (optional): Name of one or more Activity Log events to return. If not specified, all events within the range specified by `STARTDTTM` and `ENDDTTM` will be returned.
- `STARTDTTM` (optional): Start time after which events are to be returned, in the form

  `YYYY-MM-DD HH:MM:SS`

  If not specified, all events up to and including `ENDDTTM` will be returned.
- `ENDDTTM` (optional): End time prior to which events are to be returned, in the form:

  `YYYY-MM-DD HH:MM:SS`

  IF NOT SPECIFIED, ALL EVENTS STARTING AT `STARTDTTM` THROUGH THE CURRENT TIME WILL BE RETURNED.
- `NEXTKEY`: Returned with a specific value that is greater than 0, which can be used in the next call as the `NEXTKEY` value.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the events that occurred during the period requested in the `DETAILS` element block.
- `FAIL`: Returns no error message in the `ERRMSG` element in the `DETAILS` element block.
- `NEXTKEY`: Returned with a value of -1 if there are no more events to return.

Example

Use **GetEventHistory** to retrieve the events that occurred between the dates specified in `STARTDTTM` and `ENDDTTM`:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetEventHistory" num="1" dateformat="tzoffset">
       <PARAMS>
          <STARTDTTM>2016-10-01 01:01:01</STARTDTTM>
          <ENDDTTM>2016-10-30 13:04:11</ENDDTTM>
       </PARAMS>
    </COMMAND>
```

```
</NETBOX-API>
```

Successful response to the **GetEventHistory** call returns the events that occurred during the period between STARTDTTM and ENDDTTM listed in CSV format:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetEventHistory" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
LogID, Time, EventName, Type, Reason
1842,"2016-03-17 12:08:39","Motion detected at the Cafe","Event
     activated",,
1843,"2016-03-17 12:08:39","Motion detected at the Cafe","Event normal",,
1845,"2016-03-17 12:08:39","Motion detected at the Cafe","Event
     activated",,
1846,"2016-03-17 12:08:39","Motion detected at the Cafe","Event normal",,
1850,"2016-03-17 12:09:12","Motion detected at the Cafe","Event
     activated",,
1851,"2016-03-17 12:09:12","Motion detected at the Cafe","Event normal",,
1853,"2016-03-17 12:09:12","Motion detected at the Cafe","Event
     activated",,
1854,"2016-03-17 12:09:12","Motion detected at the Cafe","Event normal",,
1858,"2016-03-17 12:09:18","Motion detected at the Cafe","Event
     activated",,
1859,"2016-03-17 12:09:18","Motion detected at the Cafe","Event normal",,
1861,"2016-03-17 12:09:18","Motion detected at the Cafe","Event
     activated",,
1862,"2016-03-17 12:09:18","Motion detected at the Cafe","Event normal",,
1866,"2016-03-17 12:09:29","Motion detected at the Cafe","Event
     activated",,
1867,"2016-03-17 12:09:29","Motion detected at the Cafe","Event normal",,
1869,"2016-03-17 12:09:29","Motion detected at the Cafe","Event
     activated",,
1870,"2016-03-17 12:09:29","Motion detected at the Cafe","Event
     normal",,…
              <NEXTKEY>-1</NEXTKEY>
           </DETAILS>
     </RESPONSE>
</NETBOX>
```

This command response returns a NEXTKEY value of -1, which indicates that there are no more events that occurred in the date range to return.

GetFloors

The GetFloors command returns a list of floors.

Calling Parameters

- STARTFROMKEY (optional): The STARTROMKEY specifies the starting floor KEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of portals. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetFloors** call to return the next set of floors.

  NEXTKEY is returned with a value of -1, if there are no more portals to return. If STARTFROMKEY is omitted, all portal definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - FLOORS: Element block that contains a FLOOR element block for each floor.
  - FLOOR: Element block that contains a KEY, NAME and SEQUENCE element block for each floor.
  - KEY: Floor key.
  - NAME: Name of the floor.
  - SEQUENCE: Sequence number of the floor.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Invalid STARTFROMKEY

Example

GetFloors call to retrieve the floors using a STARTFROMKEY value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetFloors" num="1">
       <PARAMS>
          <STARTFROMKEY>0</STARTFROMKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the GetFloors call that provides a list of floors:

```
<NETBOX sessionid="1949792770">
    <RESPONSE command="GetFloors" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <FLOORS>
                <FLOOR>
                   <NAME>Penthouse</NAME>
```

```
                  <KEY>1</KEY>
                  <SEQUENCE>0</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>5</NAME>
                  <KEY>2</KEY>
                  <SEQUENCE>1</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>4</NAME>
                  <KEY>3</KEY>
                  <SEQUENCE>2</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>3</NAME>
                  <KEY>4</KEY>
                  <SEQUENCE>3</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>2</NAME>
                  <KEY>5</KEY>
                  <SEQUENCE>4</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>2R</NAME>
                  <KEY>6</KEY>
                  <SEQUENCE>5</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>1</NAME>
                  <KEY>7</KEY>
                  <SEQUENCE>6</KEY>
               </FLOOR>
               <FLOOR>
                  <NAME>Garage</NAME>
                  <KEY>8</KEY>
                  <SEQUENCE>7</KEY>
               </FLOOR>
            </FLOORS>
            <NEXTKEY>-1</NEXTKEY>
         </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns a `NEXTKEY` value of -1 which indicates that there are no more floors to return.

GetHoliday

The **GetHoliday** command returns a single holiday definition.

Calling Parameters

- HOLIDAYKEY: The holiday key.

  Use **GetHolidays** to retrieve the list of a list of HOLIDAYS key values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block for multiple holiday definitions:
  - HOLIDAY: ELEMENT BLOCK THAT CONTAINS THE HOLIDAY DEFINITION.
    - NAME: NAME OF THE HOLIDAY.
    - HOLIDAYGROUPS: NAME OF THE HOLIDAY GROUP.
    - STARTDATE: HOLIDAY START DATE. See *Date Formats*.
    - ENDDATE: HOLIDAY END DATE. See *Date Formats*.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - MISSING HOLIDAYKEYS
  - Invalid HOLIDAYKEY
  - NOT FOUND

Example

GetHoliday call to retrieve the holiday named "Thanksgiving," associated with HOLIDAYKEY value of 1:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetHoliday" num="1">
       <PARAMS>
          <HOLIDAYKEY>1</HOLIDAYKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetHoliday** command returns the holiday definition:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetHoliday" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <HOLIDAY>
                <NAME>Thanksgiving</NAME>
                <HOLIDAYGROUPS>1</HOLIDAYGROUPS>
                <STARTDATE>2016-12-25 00:00:00</STARTDATE>
                <ENDDATE>2016-12-26 00:00:00</ENDDATE>
```

```
            </HOLIDAY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetHolidays

The **GetHolidays** command returns a list of holiday keys.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameter in the DETAILS element block for multiple holiday definitions:
  - HOLIDAYS: A list of holiday keys.
- FAIL: Returns no error message in the ERRMSG element in the DETAILS element block.

Example

GetHolidays call to retrieve the list of holiday definitions:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetHolidays" num="1">
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetHolidays** call provides a list of holiday keys:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetHolidays" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <HOLIDAYS>1,2,3</HOLIDAYS>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetOutputs

The **GetOutputs** command returns a list of outputs defined on the system.

The response from a **GetOutputs** call contains a list of OUTPUTKEYs corresponding to outputs configured on the system.

The **ActivateOutput** and **DeactivateOutput** commands request that an output be activated or deactivated by specifying its OUTPUTKEY in the command.

Calling Parameters

- STARTFROMKEY: Used in conjunction with the NEXTKEY to retrieve the next set of outputs.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: DETAILS includes:
  - OUTPUTS: Used to set OUTPUT element blocks with a NAME and OUTPUTKEY.
  - NEXTKEY: Returned with:
    - A value of -1, if there are no more outputs available in the system database.
    - A specific value greater than 0 that can be used in the next call as the STARTFROMKEY value.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Invalid STARTFROMKEY

Example

GetOutputs call to retrieve the list of keys corresponding to the outputs configured:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetOutputs" num="1">
        <PARAMS>
           <STARTFROMKEY>0</STARTFROMKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetOutputs** call returns a list of the OUTPUTKEYs:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetOutputs" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <OUTPUTS>
                 <OUTPUT>
                    <NAME>Lobby Door Lock</NAME>
```

```
                        <OUTPUTKEY>36</OUTPUTKEY>
                </OUTPUT>
                <OUTPUT>
                   <NAME>Stair 1 3rd Floor Entrance Lock</NAME>
                        <OUTPUTKEY>41</OUTPUTKEY>
                </OUTPUT>
                <OUTPUT>
                   <NAME>Output 3</NAME>
                        <OUTPUTKEY>44</OUTPUTKEY>
                </OUTPUT>
                <OUTPUT>
                   <NAME>Burg panel Arming Output</NAME>
                        <OUTPUTKEY>47</OUTPUTKEY>
                </OUTPUT>
                <NEXTKEY>-1</NEXTKEY>
           </DETAILS>
     </RESPONSE>
</NETBOX>
```

`NEXTKEY` is returned with a value of -1, indicating that there are no more output keys available.

GetPartitions

The **GetPartitions** command retrieves a list of partitions and related partition information.

GetPartitions requires *User Login Authentication*.

Calling Parameters

This command has no calling parameters.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - PARTITIONS: Element block that contains the reader definition.
  - PARTITION: Element block that contains a partition definition for each partition.
  - PARTITIONKEY: Partition key.
  - NAME: Partition name.
  - DESCRIPTION: Partition description.
  - NETBOXNAME (Global Only): Name of the NetBox.
  - NETBOXIPADDRESS (Global Only): IP address of the NetBox.
- FAIL: Returns no error message in the ERRMSG element in the DETAILS element block:

Example

GetPartitions call to request a list of partition definitions:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetPartitions" num="1">
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetPartitions** call returns a list of partition definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPartitions" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PARTITIONS>
                <PARTITION>
                    <PARTITIONKEY>1</PARTITIONKEY>
                    <NAME>Master</NAME>
                    <DESCRIPTION>The default partition</DESCRIPTION>
                </PARTITION>
                <PARTITION>
                    <PARTITIONKEY>2</PARTITIONKEY>
                    <NAME>Second Partition</NAME>
                    <DESCRIPTION></DESCRIPTION>
                </PARTITION>
                <PARTITION>
```

```
                    <PARTITIONKEY>3</PARTITIONKEY>
                    <NAME>Third Partition</NAME>
                    <DESCRIPTION></DESCRIPTION>
                </PARTITION>
            </PARTITIONS>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

Successful response to the **GetPartitions** call returns a list of partition definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPartitions" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PARTITIONS>
                <PARTITION>
                    <PARTITIONKEY>1</PARTITIONKEY>
                    <NAME>Master</NAME>
                    <DESCRIPTION>The default partition</DESCRIPTION>
                </PARTITION>
                <PARTITION>
                    <PARTITIONKEY>2</PARTITIONKEY>
                    <NAME>Sales Demo</NAME>
                    <DESCRIPTION>Sales Demo Northeast Region</DESCRIPTI
ON>
                </PARTITION>
                <PARTITION>
                    <PARTITIONKEY>4</PARTITIONKEY>
                    <NAME></NAME>
                    <DESCRIPTION>Sales Demo Northeast Region</DESCRIPTI
ON>
                </PARTITION>
                <PARTITION>
                    <PARTITIONKEY>6</PARTITIONKEY>
                    <NAME>Hi</NAME>
                    <DESCRIPTION>Sales Demo Northeast Region</DESCRIPTI
ON>
                </PARTITION>
            </PARTITIONS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetPerson

The **GetPerson** command returns information defined in a person record that matches criteria specified in the calling parameters.

Use **AddPerson** to add a person record, **ModifyPerson** to modify a person record, and **RemovePerson** to remove a person record.

Calling Parameters

- PERSONID: ID number of the person whose record information is to be returned.
- USERNAME: Username associated with a person's NetBox login account.
- PASSWORD: Password associated with NetBox login account.

  If the AUTHTYPE is SSO or LDAP, then PASSWORD is not applicable and should not be set. If AUTHTYPE is DB, then PASSWORD is required.
- ROLE: Role for a person's record when they login to the NetBox.
- AUTHTYPE: Authorization type for a person's record. Valid values are DB, LDAP, or SSO.
- ALLPARTITIONS: If TRUE, retrieves the person record from any partition. If FALSE (or no parameter is specified), the record must be in the current partition.

  The ALLPARTITIONS parameter requires *User Login Authentication*, and the account the API client uses to log in must have full system setup privileges.
- PARTITIONKEY (Global API Only): The partition to which the **GetPerson** call applies when the application has been switched to a partition using the **SwitchPartition** command or the PARTITIONKEY has been set to 0.
- ACCESSLEVELDETAILS: A value of 0 indicates access level names are to be returned. A value of 1 indicates the returned access level list is to contain detailed information. Each <ACCESSLEVEL> block will contain:
  - ACCESSLEVELNAME: Name of the access level.
  - ACTDATE: Activation date for the access level.
  - EXPDATE: EXPIRATION DATE OF THE ACCESS LEVEL.
  - AUTOREMOVE: A VALUE OF 1 INDICATES THE ACCESS LEVEL WILL BE REMOVED WHEN ITS EXPIRATION DATE HAS BEEN REACHED. A VALUE OF 0 MEANS THE ACCESS LEVEL WILL NOT BE REMOVED WHEN ITS EXPIRATION DATE HAS BEEN REACHED.
- WANTCREDENTIALID: Used to request that the system return the credential ID for every credential associated with the person. The credential ID is an alias for the actual credential number.

  As a security measure, credential IDs can be retrieved and stored in a client system in place of the encoded numbers and/or hotstamp numbers.

This will allow people to manage credentials from the client system without seeing the actual credential numbers.

Use **AddCredential** to retrieve the credential ID for a credential when it is added.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns: (1) the ID number of the person whose record is being retrieved, (2) the credential ID for each of the person's credentials if `WANTCREDENTIALID` was included in the call, and (3) the following additional parameters in the `DETAILS` element block if they are defined on your system:
  - `PERSONID`: The person's ID number.
  - `FIRSTNAME`: The person's first name.
  - `MIDDLENAME`: Middle name.
  - `LASTNAME`: The person's last name.
  - `USERNAME`: Username for a person's NetBox login account.
  - `PASSWORD`: Password associated with NetBox login account.
  - `ROLE`: Role for a person's record when they login to the NetBox.
  - `AUTHTYPE`: Authorization type for a person's record. Valid values are `DB`, `LDAP`, or `SSO`.
  - `EXPDATE`: Expiration date for the person record. See *Date Formats*.
  - `ACTDATE`: Activation date of the person record. See *Date Formats*.
  - `NOTES`: Notes field of the person record.
  - `PARTITION`: Partition of person record.
  - `UDF1` through `UDF20`: User defined fields (20).
  - `DELETED`: Specifies if the person record has been deleted (`TRUE` if deleted or `FALSE`).
  - `ACCESSLEVELS`: Element block containing one or more access levels (maximum of 32) to be associated with the person. Only the access levels currently assigned are returned, and if there are none assigned, none are returned.

    The `ACCESSLEVELS` element block contains a separate element for each `ACCESSLEVEL`.
  - `ACCESSCARDS`: Element block containing one or more credentials to be associated with the person. Only the credentials currently assigned are returned, and if there are none assigned, none are returned.

    The `ACCESSCARDS` element block contains a separate element for each credential.

  - `PICTUREURL`: The picture data file returned as text between the `PICTUREURL` elements. The data file is stored in the following directory on the system:

    /usr/local/s2/web/upload/`pics`

- `FAIL`: Returns `DETAILS` which includes the `ERRMSG`:
  - Missing PERSONID
  - AUTHTYPE can only contain values - SSO, DB or LDAP.
  - The Role: full system setups does not exist.
  - Duplicate person ID: '6'.
  - ROLE is a mandatory field for AddPerson.
  - PASSWORD is a mandatory field when AUTHTYPE value is either DB or LDAP.
  - Missing ROLE.
  - Missing AUTHTYPE.

Example

GetPerson call to request information for a person record with a person ID of 21001:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetPerson" num="1">
       <PARAMS>
       <PERSONID>21001</PERSONID>
       <USERNAME>frank</USERNAME>
       <PASSWORD>doug</PASSWORD>
       <ROLE>full system setup</ROLE>
       <AUTHTYPE>DB</AUTHTYPE>
       <ALLPARTITIONS></ALLPARTITIONS>
       <ACCESSLEVELDETAILS>0</ACCESSLEVELDETAILS>
       <WANTCREDENTIALID>1</WANTCREDENTIALID>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetPerson** command response returns the person record definition:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPerson" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
          <PERSONID>21001</PERSONID>
          <FIRSTNAME>Isaac</FIRSTNAME>
          <MIDDLENAME>P</MIDDLENAME>
          <LASTNAME>Johnson</LASTNAME>
          <USERNAME>isaac</USERNAME>
          <PASSWORD>frank</PASSWORD>
          <ROLE>Master partition monitor</ROLE>
          <PARTITION>Master</PARTITION>
          <EXPDATE>2024-01-01 00:00:00</EXPDATE>
          <ACTDATE>2016-09-11 00:00:00</ACTDATE>
          <NOTES>Isaac drives a motorcycle</NOTES>
          <UDF1>Added User Defined Field 1</UDF1>
          <UDF2>Added User Defined Field 2</UDF2>
```

```
          <UDF3>Added User Defined Field 3</UDF3>
          <UDF4>Added User Defined Field 4</UDF4>
          <UDF5>Added User Defined Field 5</UDF5>
          <UDF6>Added User Defined Field 6</UDF6>
          <UDF7>Added User Defined Field 7</UDF7>
          <UDF8>Added User Defined Field 8</UDF8>
          <UDF9>Added User Defined Field 9</UDF9>
          <UDF10>Added User Defined Field 10</UDF10>
          <PICTUREURL>isaac_johnson.jpg</PICTUREURL>
          <DELETED>FALSE</DELETED>
          <ACCESSLEVELS>
             <ACCESSLEVEL>My NetBox access level 1</ACCESSLEVEL>
             <ACCESSLEVEL>My NetBox access level 2</ACCESSLEVEL>
          </ACCESSLEVELS>
          <ACCESSCARDS>
             <ACCESSCARD>
                <ENCODEDNUM>34567</ENCODEDNUM>
                <HOTSTAMP>34567</HOTSTAMP>
                <CREDENTIALID>3</CREDENTIALID>
                <CARDFORMAT>26 bit Wiegand</CARDFORMAT>
                <DISABLED>0</DISABLED>
                <CARDSTATUS>Active</CARDSTATUS>
                <CARDEXPDATE></CARDEXPDATE>
             </ACCESSCARD>
          </ACCESSCARDS>
          </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetPicture

The **GetPicture** command returns a Base64 encoded string representing a person's picture.

Calling Parameters

- `PERSONID`: Person ID of the person whose picture is being retrieved.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the following parameters in the `DETAILS` element block:
  - `PERSONID`: Person ID.
  - `PICTUREURL`: File name of the picture in the format: *name.jpg*.
  - `LASTNAME`: Last name of the person.
  - `FIRSTNAME`: First name of the person.
  - `LASTMOD`: Last time the picture was modified in `YYYY-MM-DD HH:MM:SS` format.
  - `PICTURE`: Base64 encoded string representing the person's picture.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - No picture URL for this person ID
  - Picture file does not exist
  - NOT FOUND
  - Missing PERSONID
  - Person picture image file size exceeds maximum to be returned/exported
  - Error reading picture file

Example

GetPicture call to retrieve the Base64-encoded picture file in the person record with ID number 1006:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetPicture" num="1">
       <PARAMS>
          <PERSONID>1006</PERSONID>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetPicture** call provides the Base64-encoded string:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPicture" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <PERSONID>1006</PERSONID>
```

```
            <PICTUREURL>sam ellis.jpg</PICTUREURL>
            <LASTNAME>Sam</LASTNAME>
            <FIRSTNAME>Ellis</FIRSTNAME>
            <PICTURE>/9j/4AAQSkZJRgABAQEAtAC0AAD/7RMcUGhvdG
                                .
                                .
                                .
                        U68tqfLOkxet9Xi9X+84Lz/wBag5YO/uUv/9k=
            </PICTURE>
          </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetPortalGroup

The **GetPortalGroup** command retrieves information about a single portal group.

Calling Parameters

- PORTALGROUPKEY: A portal group key that corresponds to a portal group.

  Use **GetPortalGroups** to retrieve the list of PORTALGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - PORTALGROUP: Element block that contains the details of the portal group.
    - PORTALGROUPKEY: Key that corresponds to a portal group.
    - NAME: Name of the portal group.
    - DESCRIPTION: Description of the portal group.
    - PORTALS: Element block that contains the portal definition.
    - PORTALKEY: Key that corresponds to the portal.
    - NAME: Name of the portal.
    - UNLOCKTIMESPECGROUPKEY: Time spec group key for unlocking portals in this group.
    - THREATLEVELGROUPKEY: Threat level group key corresponding to the threat level group assigned to the portal group.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS block:
  - Missing PORTALGROUPKEY
  - INVALID PORTALGROUPKEY
  - NOT FOUND

If the key is not a valid group, then the DETAILS block will contain no PORTALGROUP elements.

Example

GetPortalGroup call to retrieve the details for the portal group named, "Gym Doors", which corresponds to the PORTALGROUPKEY value of 33:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetPortalGroup" num="1">
       <PARAMS>
          <PORTALGROUPKEY>33</PORTALGROUPKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetPortalGroup** call that provides a list of portal definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetPortalGroup" num="1">
```

```
      <CODE>SUCCESS</CODE>
        <DETAILS>
            <PORTALGROUP>
                <PORTALGROUPKEY>33</PORTALGROUPKEY>
                <NAME>Gym Doors</NAME>
                <DESCRIPTION></DESCRIPTION>
                <PORTALS>
                    <PORTAL>
                        <PORTALKEY>1</PORTALKEY>
                        <NAME>2nd Floor Delivery Area</NAME>
                    </PORTAL>
                    <PORTAL>
                        <PORTALKEY>4</PORTALKEY>
                        <NAME>Lobby Door</NAME>
                    </PORTAL>
                    <PORTAL>
                        <PORTALKEY>5</PORTALKEY>
                        <NAME>Stair 1 3rd Floor Entrance</NAME>
                    </PORTAL>
                </PORTALS>
                <UNLOCKTIMESPECGROUPKEY>2</UNLOCKTIMESPECGROUPKEY>
                <THREATLEVELGROUPKEY></THREATLEVELGROUPKEY>
            </PORTALGROUP>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetPortalGroups

The **GetPortalGroups** command returns a list of portal group definitions, which includes PORTALGROUPKEYs corresponding to portal groups.

Calling Parameters

- STARTFROMKEY: The STARTROMKEY specifies the starting PORTALGROUPKEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of time spec groups. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetPortalGroups** call to return the next set of portal groups.

  NEXTKEY is returned with a value of -1, if there are no more portal groups to return.

  If STARTFROMKEY is omitted, all portal group definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - PORTALGROUPS: Element block that contains a PORTALGROUP element block for each portal group.
  - PORTALGROUP: Element block that contains details of the portal group.
  - PORTALGROUPKEY: Key corresponding to a portal group.
  - NAME: Name of the portal group.
  - DESCRIPTION: Description of the portal group.
  - PORTALS: Element block that contains a list of PORTALKEYS.
  - PORTAL: Element block that contains the details of the portal.
  - PORTALKEY: Key that corresponds to a portal.
  - NAME: Name of a portal.
  - UNLOCKTIMESPECGROUPKEY: Time spec group key for unlocking portals in this group.
  - THREATLEVELGROUPKEY: Threat level group key corresponding to the threat level group assigned to the portal group.
  - NEXTKEY: Returned with a value of -1, if there are no more time spec groups to return, or a specific value > 0 that can be used in the next call as the STARTFROMKEY value.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Invalid STARTFROMKEY

Example

GetPortalGroups call includes a `STARTFROMKEY` value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetPortalGroups" num="1">
        <PARAMS>
           <STARTFROMKEY>0</STARTFROMKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetPortalGroups** call that provides a list of portal group definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetPortalGroups" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
           <PORTALGROUPS>
              <PORTALGROUP>
                 <PORTALGROUPKEY>33</PORTALGROUPKEY>
                 <NAME>Gym Doors</NAME>
                 <DESCRIPTION></DESCRIPTION>
                 <PORTALS>
                    <PORTAL>
                          <PORTALKEY>1</PORTALKEY>
                          <NAME>2nd Floor Delivery Area</NAME>
                    </PORTAL>
                    <PORTAL>
                          <PORTALKEY>4</PORTALKEY>
                          <NAME>Lobby Door</NAME>
                    </PORTAL>
                    <PORTAL>
                          <PORTALKEY>5</PORTALKEY>
                          <NAME>Stair 1 3rd Floor Entrance</NAME>
                    </PORTAL>
                          </PORTALS>
                 <UNLOCKTIMESPECGROUPKEY>2</UNLOCKTIMESPECGROUPKEY>
                 <THREATLEVELGROUPKEY></THREATLEVELGROUPKEY>
              </PORTALGROUP>
              <PORTALGROUP>
                 <PORTALGROUPKEY>34</PORTALGROUPKEY>
                 <NAME>Entrance Doors</NAME>
                 <DESCRIPTION></DESCRIPTION>
                 <PORTALS>
                    <PORTAL>
                          <PORTALKEY>1</PORTALKEY>
                          <NAME>2nd Floor Delivery Area</NAME>
                    </PORTAL>
                 </PORTALS>
                 <UNLOCKTIMESPECGROUPKEY>2</UNLOCKTIMESPECGROUPKEY>
                 <THREATLEVELGROUPKEY></THREATLEVELGROUPKEY>
              </PORTALGROUP>
              <PORTALGROUP>
                 <PORTALGROUPKEY>42</PORTALGROUPKEY>
                 <NAME>all doors</NAME>
                 DESCRIPTION></DESCRIPTION>
                 <PORTALS>
```

```
                    <PORTAL>
                          <PORTALKEY>1</PORTALKEY>
                          <NAME>2nd Floor Delivery Area</NAME>
                    </PORTAL>
                    <PORTAL>
                          <PORTALKEY>4</PORTALKEY>
                          <NAME>Lobby Door</NAME>
                    </PORTAL>
                    <PORTAL>
                          <PORTALKEY>5</PORTALKEY>
                          <NAME>Stair 1 3rd Floor Entrance</NAME>
                    </PORTAL>
                 </PORTALS>
                 <UNLOCKTIMESPECGROUPKEY>2</UNLOCKTIMESPECGROUPKEY>
                 <THREATLEVELGROUPKEY></THREATLEVELGROUPKEY>
              </PORTALGROUP>
           </PORTALGROUPS>
           <NEXTKEY>-1</NEXTKEY>
           </DETAILS>
     </RESPONSE>
</NETBOX>
```

This command is returned with a NEXTKEY value of -1 which indicates that there are no more portal groups to return.

GetPortals

The **GetPortals** command returns a list of portals and associated card readers.

Calling Parameters

- STARTFROMKEY (optional): The STARTROMKEY specifies the starting PORTALKEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of portals. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetPortals** call to return the next set of portals.

  NEXTKEY is returned with a value of -1, if there are no more portals to return. If STARTFROMKEY is omitted, all portal definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block for multiple readers:
  - PORTALS: Element block that contains a PORTAL element block for each time spec.
  - PORTAL: Element block that contains the portal definition.
  - NAME: Name of the portal.
  - PORTALKEY: Key corresponding to a portal.
  - READERS: Element block that contains the reader definitions.
  - READER: Element block that contains a reader definition.
  - READERKEY: Key that corresponds to the reader.
  - NAME: Name of the reader.
  - DESCRIPTION: Description of the reader.
  - PORTAL ORDER: Indicates whether the reader is in the first or the second position in a portal. The first position is used as the incoming reader; the second position is optional, and is used as the outgoing reader in a dual reader door when it exists.

    Specify either 1 or 2.

  Note: The first and second positions are not guaranteed as incoming and outgoing as described above.

- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block.
  - NOT FOUND
  - Invalid STARTFROMKEY

Example

GetPortals call to retrieve the portals using a `STARTFROMKEY` value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetPortals" num="1">
        <PARAMS>
           <STARTFROMKEY>0</STARTFROMKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetPortals** call that provides a list of portals and reader definitions and includes their `READERKEY`s and `PORTALKEY`s:

```
<NETBOX sessionid="1949792770">
     <RESPONSE command="GetPortals" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
              <PORTALS>
                 <PORTAL>
                 <NAME>2nd Floor Delivery Area</NAME>
                 <PORTALKEY>1</PORTALKEY>
                    <READERS>
                          <READER>
                          <READERKEY>50</READERKEY>
                          <NAME>2nd Floor Delivery Area</NAME>
                          <PORTALORDER>1</PORTALORDER>
                    </READER>
                    </READERS>
                 </PORTAL>
                 <PORTAL>
                 <NAME>Lobby Door</NAME>
                 <PORTALKEY>4</PORTALKEY>
                    <READERS>
                          <READER>
                          <READERKEY>16</READERKEY>
                          <NAME>Lobby Door</NAME>
                          <PORTALORDER>1</PORTALORDER>
                    </READER>
                    </READERS>
                 </PORTAL>
                 <PORTAL>
                 <NAME>Stair 1 3rd Floor Emp. Entrance</NAME>
                 <PORTALKEY>5</PORTALKEY>
                    <READERS>
                          <READER>
                          <READERKEY>19</READERKEY>
                          <NAME>Stair 1 3rd Floor Entrance</NAME>
                          <PORTALORDER>1</PORTALORDER>
                          </READER>
                    </READERS>
                 </PORTAL>
              </PORTALS>
              <NEXTKEY>-1</NEXTKEY>
           </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns a `NEXTKEY` value of -1 which indicates that there are no more portals to return.

GetReader

The **GetReader** command returns information about a single reader.

Calling Parameters

- READERKEY: The reader key.

  Use GetReaders to retrieve the list of READERKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - READER: Element block that contains the reader definition.
  - READERKEY: Key corresponding to the reader.
  - NAME: Name of the reader.
  - DESCRIPTION: Reader description.
- FAIL: Returns an error messag in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Missing READERKEY
  - INVALID READERKEY

If the key is not valid, then the DETAILS block will contain no READER element.

Example

GetReader call to retrieve the reader named, "Lobby Door," associated with a READERKEY with the value of 16:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetReader" num="1">
       <PARAMS>
          <READERKEY>16</READERKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetReader** call that provides the definition of the requested reader:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetReader" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <READER>
                <READERKEY>16</READERKEY>
                <NAME>Lobby Door</NAME>
                <DESCRIPTION></DESCRIPTION>
             </READER>
          </DETAILS>
```

```
    </RESPONSE>
</NETBOX>
```

GetReaders

The **GetReaders** command returns a list of reader definitions, which include READERKEYs corresponding to readers.

Calling Parameters

- STARTFROMKEY (optional): The STARTROMKEY specifies the starting READERKEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of readers. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetReaders** call to return the next set of readers.

  NEXTKEY is returned with a value of -1, if there are no more readers to return. If STARTFROMKEY is omitted, all reader definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block for multiple readers:
  - READERS: Element block that contains a READER element block for each reader.
  - READER: Element block that contains the reader definition.
  - NAME: Name of the reader.
  - DESCRIPTION: Description of the reader.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - INVALID STARTFROMKEY

Example

GetReaders call to retrieve the readers using a STARTFROMKEY value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetReaders" num="1">
        <PARAMS>
           <STARTFROMKEY>0</STARTFROMKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetReaders** call that provides a list of reader definitions and includes their READERKEYs:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetReaders" num="1">
        <CODE>SUCCESS</CODE>
           <DETAILS>
```

```
            <READERS>
               <READER>
                  <READERKEY>16</READERKEY>
                  <NAME>Lobby Door</NAME>
                  <DESCRIPTION></DESCRIPTION>
               </READER>
               <READER>
                  <READERKEY>19</READERKEY>
                  <NAME>Stair 1 3rd Floor Emp. Entrance</NAME>
                  <DESCRIPTION></DESCRIPTION>
               </READER>
               <READER>
                  <READERKEY>50</READERKEY>
                  <NAME>2nd Floor Delivery Area</NAME>
                  <DESCRIPTION></DESCRIPTION>
               </READER>
               <READER>
                  <READERKEY>53</READERKEY>
                  <NAME>Elevator Test Reader</NAME>
                  <DESCRIPTION></DESCRIPTION>
               </READER>
               <READER>
                  <READERKEY>196</READERKEY>
                  <NAME>Reader1 on CASI M5</NAME>
                  <DESCRIPTION></DESCRIPTION>
               </READER>
               <READER>
                  <READERKEY>199</READERKEY>
                  <NAME>Reader2 on CASI M5</NAME>
                  <DESCRIPTION></DESCRIPTION>
               </READER>
               </READERS>
         <NEXTKEY>-1</NEXTKEY>
         </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command response returns a NEXTKEY value of -1 which indicates that there are no more readers to return.

GetReaderGroup

The **GetReaderGroup** command returns a single reader group.

Calling Parameters

- READERGROUPKEY: A reader group key that corresponds to a time spec group.

  Use **GetReaderGroups** to retrieve the list of READERGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - READERGROUP: Element block that contains the details of the reader group.
  - READERGROUPKEY: Key that corresponds to a reader group.
  - NAME: Name of the reader group.
  - DESCRIPTION: Description of the reader group.
  - READERS: Element block that contains a list of READERS.
  - READERKEY: Key that corresponds to a reader.
  - NAME: Name of the reader.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Missing READERGROUPKEY
  - INVALID READERGROUPKEY

Example

GetReaderGroup call to retrieve the details for the reader group named "All Readers Group", by supplying the READERGROUPKEY value of 21:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetReaderGroup" num="21">
       <PARAMS>
          <READERGROUPKEY>21</READERGROUPKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetReaderGroup** call that provides the definition of the requested reader group:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetReaderGroup" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <READERGROUP>
                <READERGROUPKEY>21</READERGROUPKEY>
                <NAME>All Readers Group</NAME>
```

```
                <DESCRIPTION></DESCRIPTION>
                <READERS>
                    <READER>
                        <READERKEY>16</READERKEY>
                        <NAME>Lobby Door</NAME>
                    </READER>
                    <READER>
                        <READERKEY>50</READERKEY>
                        <NAME>2nd Floor Delivery Area</NAME>
                    </READER>
                    <READER>
                        <READERKEY>19</READERKEY>
                        <NAME>Stair 1 3rd Floor Entrance</NAME>
                    </READER>
                    <READER>
                        <READERKEY>53</READERKEY>
                        <NAME>Elevator Test Reader</NAME>
                    </READER>
                </READERS>
            </READERGROUP>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetReaderGroups

The **GetReaderGroups** command returns a list of reader group definitions, which includes READERGROUPKEYs corresponding to reader groups.

Calling Parameters

- STARTFROMKEY: The STARTFROMKEY specifies the starting READERGROUPKEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of time spec groups. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetReaderGroups** call to return the next set of reader groups.

  NEXTKEY is returned with a value of -1, if there are no more reader groups to return.

  If STARTFROMKEY is omitted, all reader group definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - READERGROUPS: Element block that contains a READERGROUP element block for each reader group.
  - READERGROUP: Element block that contains details of the reader group.
  - READERGROUPKEY: Key corresponding to a reader group.
  - NAME: Name of the reader group.
  - DESCRIPTION: Description of the reader group.
  - READERS: Element block that contains a READER block for each reader in the reader group.
  - READER: Element block that contains the READERKEY and the reader name.
  - READERKEY: Key that corresponds to a reader.
  - NAME: Name of a Reader.
  - NEXTKEY: Returned with a value of -1, if there are no more reader groups to return, or a specific value > 0 that can be used in the next call as the STARTFROMKEY value.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND

Example

GetReaderGroups call that includes a STARTFROMKEY value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetReaderGroups" num="1">
       <STARTFROMKEY>0</STARTFROMKEY>
```

```
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetReaderGroups** call that provides a list of reader group definitions, which includes their READERKEYs and READERGROUPKEYs:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetReaderGroups" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <READERGROUPS>
                <READERGROUP>
                    <READERGROUPKEY>21</READERGROUPKEY>
                    <NAME>All Readers Group</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <READERS>
                        <READER>
                            <READERKEY>16</READERKEY>
                            <NAME>Lobby Door</NAME>
                        </READER>
                        <READER>
                            <READERKEY>50</READERKEY>
                            <NAME>2nd Floor Delivery Area</NAME>
                        </READER>
                        <READER>
                            <READERKEY>19</READERKEY>
                            <NAME>Stair 1 3rd Floor Entrance</NAME>
                        </READER>
                        <READER>
                            <READERKEY>53</READERKEY>
                            <NAME>Elevator Test Reader</NAME>
                        </READER>
                    </READERS>
                </READERGROUP>
                <READERGROUP>
                    <READERGROUPKEY>28</READERGROUPKEY>
                    <NAME>2nd Floor Readers Group</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <READERS>
                        <READER>
                            <READERKEY>50</READERKEY>
                            <NAME>2nd Floor Delivery Area</NAME>
                        </READER>
                    </READERS>
                </READERGROUP>
                <READERGROUP>
                    <READERGROUPKEY>43</READERGROUPKEY>
                    <NAME>Waterville</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <READERS>
                        <READER>
                            <READERKEY>53</READERKEY>
                            <NAME>Elevator Test Reader</NAME>
                        </READER>
                        <READER>
                            <READERKEY>19</READERKEY>
                            <NAME>Stair 1 3rd Floor Entrance</NAME>
```

```
                        </READER>
                    </READERS>
                </READERGROUP>
                <READERGROUP>
                    <READERGROUPKEY>45</READERGROUPKEY>
                    <NAME>Common Area Readers</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <READERS>
                        <READER>
                            <READERKEY>50</READERKEY>
                            <NAME>2nd Floor Delivery Area</NAME>
                        </READER>
                        <READER>
                            <READERKEY>196</READERKEY>
                            <NAME>Reader1 on CASI M5</NAME>
                        </READER>
                        <READER>
                            <READERKEY>199</READERKEY>
                            <NAME>Reader2 on CASI M5</NAME>
                        </READER>
                        <READER>
                            <READERKEY>19</READERKEY>
                            <NAME>Stair 1 3rd Floor Entrance</NAME>
                        </READER>
                        <READER>
                            <READERKEY>16</READERKEY>
                            <NAME>Lobby Door</NAME>
                       </READER>
                    </READERS>
                </READERGROUP>
            </READERGROUPS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command is returned with a `NEXTKEY` value of -1, which indicates that there are no more reader groups to return.

GetTimeSpec

The **GetTimeSpec** command returns information about a single time spec. This command returns a time spec associated with the TIMESPECKEY that is requested for in the input XML. If the TIMESPECKEY is not available in the current partition, a response of NOT FOUND is returned. If the TIMESPECKEY of "Always" or "Never" Timespec is requested, it will be returned irrespective of any partition id.

Calling Parameters

- `TIMESPECKEY`: A time spec key.

  Use **GetTimeSpecs** to retrieve the list of `TIMESPECKEY` values.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the following parameters in the `DETAILS` element block:
  - `TIMESPEC`: Element block that contains the time spec definition.
  - `TIMESPECKEY`: Key corresponding to the time spec.
  - `NAME`: Name of the time spec.
  - `DESCRIPTION`: Description of the time spec.
  - `STARTTIME`: Start Time in `HH:MM` format.
  - `ENDTIME`: End Time in `HH:MM` format.
  - Days of the week included, 1 for `TRUE` and 0 for `FALSE`, for each of:

    ```
    MONDAY
    TUESDAY
    WEDNESDAY
    THURSDAY
    FRIDAY
    SATURDAY
    SUNDAY
    ```

  - `HOLIDAYGROUPS`: Holiday group numbers (1,2,3,4,5,6,7,8 separated by commas).
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - NOT FOUND
  - Missing TIMESPECKEY
  - Invalid TIMESPECKEY

Example

GetTimeSpec call to retrieve the time spec named "M-F 8AM-5PM Visitor Hours," associated with `TIMESPECKEY` value of 8:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetTimeSpec" num="1">
```

```
        <PARAMS>
           <TIMESPECKEY>8</TIMESPECKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetTimeSpec** call that provides the definition of the requested time spec:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetTimeSpec" num="1">
     <CODE>SUCCESS</CODE>
        <DETAILS>
           <TIMESPEC>
           <TIMESPECKEY>8</TIMESPECKEY>
              <NAME>M-F 8AM-5PM Visitor Hours</NAME>
              <DESCRIPTION></DESCRIPTION>
              <STARTTIME>08:00</STARTTIME>
              <ENDTIME>17:00</ENDTIME>
              <MONDAY>TRUE</MONDAY>
              <TUESDAY>TRUE</TUESDAY>
              <WEDNESDAY>TRUE</WEDNESDAY>
              <THURSDAY>TRUE</THURSDAY>
              <FRIDAY>TRUE</FRIDAY>
              <SATURDAY>FALSE</SATURDAY>
              <SUNDAY>FALSE</SUNDAY>
              <HOLIDAYGROUPS></HOLIDAYGROUPS>
           </TIMESPEC>
        </DETAILS>
     </RESPONSE>
</NETBOX>
```

GetTimeSpecs

The **GetTimeSpecs** command returns a list of time spec definitions, which includes TIMESPECKEYs corresponding to time specs.

Calling Parameters

- STARTFROMKEY (optional): The STARTROMKEY specifies the starting TIMESPECKEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of time specs. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetTimeSpecs** call to return the next set of time specs.

  NEXTKEY is returned with a value of -1, if there are no more time specs to return.

  If STARTFROMKEY is omitted, all time spec definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block for multiple time specs:
  - TIMESPECS: Element block that contains a TIMESPEC element block for each time spec.
  - TIMESPEC: Element block contains the time spec definition.
  - TIMESPECKEY: Key corresponding to a time spec.
  - NAME: Name of the time spec.
  - DESCRIPTION: Description of the time spec.
  - STARTTIME: Start Time in HH:MM format.
  - ENDTIME: End Time in HH:MM format.
  - Days of the week included, 1 for TRUE and 0 for FALSE, for each of:

    ```
    MONDAY
    TUESDAY
    WEDNESDAY
    THURSDAY
    FRIDAY
    SATURDAY
    SUNDAY
    ```

  - HOLIDAYGROUPS: Holiday group numbers (1,2,3,4,5,6,7,8 separated by commas).
  - NEXTKEY: Returned with a value of -1, if there are no more time specs to return, or a specific value > 0 that can be used in the next call as the STARTFROMKEY value.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:

- NOT FOUND

Example

GetTimeSpecs call to retrieve the time specs using a `STARTFROMKEY` value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetTimeSpecs" num="1">
        <PARAMS>
           <STARTFROMKEY>0</STARTFROMKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **GetTimeSpecs** call that provides a list of time spec definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetTimeSpecs" num="1">
     <CODE>SUCCESS</CODE>
     <DETAILS>
        <TIMESPECS>
           <TIMESPEC>
              <TIMESPECKEY>8</TIMESPECKEY>
                 <NAME>M-F 8AM-5PM Visitor Hours</NAME>
                 <DESCRIPTION></DESCRIPTION>
                 <STARTTIME>08:00</STARTTIME>
                 <ENDTIME>17:00</ENDTIME>
                 <MONDAY>TRUE</MONDAY>
                 <TUESDAY>TRUE</TUESDAY>
                 <WEDNESDAY>TRUE</WEDNESDAY>
                 <THURSDAY>TRUE</THURSDAY>
                 <FRIDAY>TRUE</FRIDAY>
                 <SATURDAY>FALSE</SATURDAY>
                 <SUNDAY>FALSE</SUNDAY>
                 <HOLIDAYGROUPS></HOLIDAYGROUPS>
           </TIMESPEC>
           <TIMESPEC>
              <TIMESPECKEY>6</TIMESPECKEY>
                 <NAME>All Week 6AM-7PM Patriot</NAME>
                 <DESCRIPTION></DESCRIPTION>
                 <STARTTIME>06:00</STARTTIME>
                 <ENDTIME>19:00</ENDTIME>
                 <MONDAY>TRUE</MONDAY>
                 <TUESDAY>TRUE</TUESDAY>
                 <WEDNESDAY>TRUE</WEDNESDAY>
                 <THURSDAY>TRUE</THURSDAY>
                 <FRIDAY>TRUE</FRIDAY>
                 <SATURDAY>TRUE</SATURDAY>
                 <SUNDAY>TRUE</SUNDAY>
                 <HOLIDAYGROUPS>1,2,3</HOLIDAYGROUPS>
           </TIMESPEC>
        </TIMESPECS>
        <NEXTKEY>-1</NEXTKEY>
     </DETAILS>
     </RESPONSE>
</NETBOX>
```

This command response returns a `NEXTKEY` value of -1, which indicates that there are no more time specs to return.

GetTimeSpecGroup

The **GetTimeSpecGroup** command returns a single time spec group.

Calling Parameters

- TIMESPECGROUPKEY: A time spec group key that corresponds to a time spec group.

  Use **GetTimeSpecGroups** to retrieve the list of TIMESPECGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - TIMESPECGROUP: Element block that contains the details of the time spec group.
  - TIMESPECGROUPKEY: Key that corresponds to a time spec group.
  - NAME: Name of the time spec group.
  - DESCRIPTION: Description of the time spec group.
  - TIMESPECKEYS: Element blocks that contains a list of TIMESPECKEYs.
  - TIMESPECKEY: Key that corresponds to a time spec.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS block:
  - NOT FOUND
  - Missing TIMESPECGROUPKEY
  - INVALID TIMESPECGROUPKEY

Example

GetTimeSpecGroup call to retrieve the details for the time spec group named "Always," by supplying the TIMESPECGROUPKEY value of 1:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetTimeSpecGroup" num="1">
        <PARAMS>
           <TIMESPECGROUPKEY>1</TIMESPECGROUPKEY>
        </PARAMS>
     </COMMAND>
<NETBOX-API>
```

Successful response to the **GetTimeSpecGroup** call that provides a list of time spec group definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="GetTimeSpecGroup" num="1">
     <CODE>SUCCESS</CODE>
        <DETAILS>
           <TIMESPECGROUP>
```

```
            <TIMESPECGROUPKEY>1</TIMESPECGROUPKEY>
               <NAME>Always</NAME>
               <DESCRIPTION></DESCRIPTION>
                  <TIMESPECKEYS>
                        <TIMESPECKEY>1</TIMESPECKEY>
                  </TIMESPECKEYS>
            </TIMESPECGROUP>
      </DETAILS>
   </RESPONSE>
</NETBOX>
```

GetTimeSpecGroups

The **GetTimeSpecGroups** command returns a list of time spec group definitions, which includes TIMESPECGROUPKEYs corresponding to time spec groups.

Calling Parameters

- STARTFROMKEY: The STARTROMKEY specifies the starting TIMESPECGROUPKEY.

  STARTFROMKEY is used in conjunction with NEXTKEY to retrieve the next set of time spec groups. If NEXTKEY is returned with a value greater than 0, a STARTFROMKEY parameter value can be passed in the next **GetTimeSpecGroups** call to return the next set of time spec groups.

  NEXTKEY is returned with a value of -1, if there are no more time spec groups to return.

  If STARTFROMKEY is omitted, all time spec group definitions are returned.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameters in the DETAILS element block:
  - TIMESPECGROUPS: Element block that contains a TIMESPECGROUP element block for each time spec group.
  - TIMESPECGROUP: Element block that contains details of the time spec group.
  - TIMESPECGROUPKEY: Key corresponding to a time spec group.
  - NAME: Name of the time spec group.
  - DESCRIPTION: Description of the time spec group.
  - TIMESPECKEYS: Element block that contains a list of TIMESPECKEYS.
  - TIMESPECKEY: Key that corresponds to a time spec.
  - NEXTKEY: Returned with a value of -1 if there are no more time spec groups to return, or a specific value > 0 that can be used in the next call as the STARTFROMKEY value.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Invalid STARTFROMKEY

Example

GetTimeSpecGroups call includes a STARTFROMKEY value of 0:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetTimeSpecGroups" num="1">
       <PARAMS>
          <STARTFROMKEY>0</STARTFROMKEY>
```

```
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetTimeSpecGroups** call that provides a list of time spec group definitions:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetTimeSpecGroups" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <TIMESPECGROUPS>
                <TIMESPECGROUP>
                    <TIMESPECGROUPKEY>1</TIMESPECGROUPKEY>
                    <NAME>Always</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <TIMESPECKEYS>
                        <TIMESPECKEY>1</TIMESPECKEY>
                    </TIMESPECKEYS>
                </TIMESPECGROUP>
                <TIMESPECGROUP>
                    <TIMESPECGROUPKEY>2</TIMESPECGROUPKEY>
                    <NAME>Never</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <TIMESPECKEYS>
                        <TIMESPECKEY>2</TIMESPECKEY>
                    </TIMESPECKEYS>
                </TIMESPECGROUP>
                <TIMESPECGROUP>
                    <TIMESPECGROUPKEY>22</TIMESPECGROUPKEY>
                    <NAME>6P-10P Tuesday</NAME>
                    <DESCRIPTION></DESCRIPTION>
                    <TIMESPECKEYS>
                        <TIMESPECKEY>3</TIMESPECKEY>
                    </TIMESPECKEYS>
                </TIMESPECGROUP>
            <TIMESPECGROUPS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

This command is returned with a `NEXTKEY` value of -1 which indicates that there are no more time spec groups to return.

GetUDFLists

The **GetUDFLists** command retrieves a list of the UDF (user-defined field) value lists defined in the system.

Use **GetUDFListItems** to retrieve a list of the values in a UDF value list. Use **ModifyUDFListItems** to add, modify, or delete values in a UDF value list.

Calling Parameters

None.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the following parameter in the DETAILS element block for each UDF value list containing multiple values:
  - UDFLISTKEY: The unique key for the UDF value list.
  - NAME: The name of the UDF value list.
  - DESCRIPTION: A description of the UDF value list, if available.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND

EXAMPLE

GetUDFLists call to retrieve a list of UDF value list definitions:

```
<NETBOX-API sessionid="2129346402">
    <COMMAND name="GetUDFLists" num="1" dateformat="tzoffset">
    </COMMAND>
</NETBOX-API>
```

Successful response to the GetUDFLists call, which provides the UDF value lists:

```
<NETBOX sessionid="2129346402">
    <RESPONSE command="GetUDFLists" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <UDFLISTS>
             <UDFLIST>
                <UDFLISTKEY>1</UDFLISTKEY>
                <NAME>Managers</NAME>
                <DESCRIPTION>Division List</DESCRIPTION>
             </UDFLIST>
             <UDFLIST>
                <UDFLISTKEY>2</UDFLISTKEY>
                <NAME>Department List</NAME>
                <DESCRIPTION>Departments</DESCRIPTION>
             </UDFLIST>
          </UDFLISTS>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetUDFListItems

The **GetUDFListItems** command retrieves a list of the values in an existing UDF (user-defined field) value list

Use **GetUDFLists** to obtain the unique UDFLISTKEY for a UDF value list. Use **ModifyUDFListItems** to add, modify, or delete values in a UDF value list.

Calling Parameters

- UDFLISTKEY: The unique key for the UDF value list.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the name and unique key for the UDF value list and a list of the values it contains:
  - NAME: The name of the UDF value list.
  - UDFLISTKEY: The unique key for the UDF value list.
  - ITEMNAME The name of a value in the list.
  - ITEMKEY: The unique key for a value in the list.
  - CUSTOMKEY: A unique key assigned to a value when it was added to the list.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - NOT FOUND
  - Missing UDFLISTKEY
  - INVALID UDFLISTKEY

EXAMPLE

GetUDFListItems call to retrieve a list of the values in a UDF value list:

```
<NETBOX-API sessionid="3769758827">
    <COMMAND name="GetUDFListItems" num="1" dateformat="tzoffset">
       <PARAMS>
          <UDFLISTKEY>1</UDFLISTKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **GetUDFListItems** call that provides a list of the values in the UDF value list:

```
<NETBOX sessionid="3769758827">
    <RESPONSE command="GetUDFLists" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <UDFLISTKEY>1</UDFLISTKEY>
          <NAME>Division List</NAME>
          <LISTITEMS>
             <LISTITEM>
```

```
                <ITEMKEY>65</ITEMKEY>
                <ITEMNAME>Eastern Division</ITEMNAME>
                <CUSTOMKEY>_1</CUSTOMKEY>
             </LISTITEM>
             <LISTITEM>
                <ITEMKEY>66</ITEMKEY>
                <ITEMNAME>Western Division</ITEMNAME>
                <CUSTOMKEY>_2</CUSTOMKEY>
             </LISTITEM>
          </LISTITEMS>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

GetVirtualCredentialRequest

The **GetVirtualCredentialRequest** command retrieves a mobile credential and its assignment state.

.

Calling Parameters

- PERSONID: ID of the person associated with the credential to be added.
- CARDFORMAT: Name of the format to be used to decode the credential.

Note: PERSONID and CARDFORMAT are mandatory parameters.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the CARDFORMAT , PERSONID, STATUS of the retrieved credential in the DETAIL element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - CARDFORMAT NOT FOUND

Example

GetVirtualCredentialRequest call to request get mobile credential with a specific CARDFORMAT that is assigned to a person with PERSONID:

```
<NETBOX-API>
    <COMMAND name="GetVirtualCredentialRequest" num="1">
        <PARAMS>
            <PERSONID>123</PERSONID>
            <CARDFORMAT>LenelS2 48bit Corp 1000</CARDFORMAT>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to GetVirtualCredentialRequest call:

```
<NETBOX>
    <RESPONSE command="GetVirtualCredentialRequest" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <VIRTUALCREDENTIALREQUEST>
                <CARDFORMAT>LenelS2 48bit Corp 1000</CARDFORMAT>
                <PERSONID>123</PERSONID>
                <STATUS>Active</STATUS>
            </VIRTUALCREDENTIALREQUEST>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

InsertActivity

The InsertActivity command can be used to record Access granted and Access denied events in the Activity Log.

The command also allows the insertion of a user-defined event that displays a text string in the Activity Log. In this case, only the ACTIVITYTYPE and ACTIVITYTEXT parameters must be included. In addition, num="1" must be included after the command name.

Calling Parameters

- ACTIVITYTYPE: The activity to be inserted:
- ACCESSGRANTED
- ACCESSDENIED
- USERACTIVITY
- DETAILS: Detail that supports an ACCESSDENIED:
- DISABLED
- EXPIRED
- LOCATION
- PIN
- TIME
- UNKNOWN
- The key associated with the portal for Access Granted or Access Denied. Use **GetOutputs** to retrieve the list of PORTALKEY values. **InsertActivity** can only specify one of PORTALKEY or ELEVATORKEY.
- ELEVATORKEY: The key associated with the elevator for Access Granted or Access Denied. Use **GetOutputs** to retrieve the list of ELEVATORKEY values. **InsertActivity** can only specify one of PORTALKEY or ELEVATORKEY.
- FLOORKEY: The key associated with the floor for Access Granted with an ELEVATORKEY. Use **GetOutputs** to retrieve the list of FLOORKEY values.
- READERKEY: The key associated with reader for Access Granted or Access Denied. Use **GetOutputs** to retrieve the list of READEREY values.
- PERSONID: ID of person for whom the activity is associated.
- CARDFORMAT: Text name of the card format used as part of the activity entry.
- ENCODEDNUM: Specifies the encoded number used as part of the activity entry.
- ACTIVITYTEXT: A string of text (limit 255 chars) to be displayed with the activity entry.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Activity text is limited to {0} characters
  - One of ELEVATORKEY or PORTALKEY is required
  - ACTIVITYTYPE is required
  - ENCODEDNUM is required
  - CARDFORMAT is required
  - PERSONID is required
  - READERKEY is required
  - DETAILS is required
  - PORTALKEY is required
  - ELEVATORKEY is required
  - FLOORKEY is required
  - Invalid portal key
  - Invalid elevator key
  - Invalid floor key
  - Invalid reader key
  - Invalid person id
  - ACTIVITYTYPE must be ACCESSGRANTED, ACCESSDENIED or USERACTIVITY
  - DETAILS must be DISABLED, EXPIRED, LOCATION, PIN, TIME or UNKNOWN
  - Invalid card format and encoded number
  - Unable to insert activity
  - Invalid parameters for XXX activity.

Example

InsertActivity call for Access Granted for a person at a portal:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="InsertActivity">
        <PARAMS>
           <ACTIVITYTYPE>ACCESSGRANTED</ACTIVITYTYPE>
           <PORTALKEY>15</PORTALKEY>
           <READERKEY>23</READERKEY>
           <PERSONID>527</PERSONID>
```

```
            <CARDFORMAT>26 Bit Wiegand FC 7</CARDFORMAT>
            <ENCODEDNUM>555</ENCODEDNUM>
         </PARAMS>
  </COMMAND>
</NETBOX-API>
```

InsertActivity call for Access Denied due to time specification for a person at a portal:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="InsertActivity">
         <PARAMS>
            <ACTIVITYTYPE>ACCESSDENIED</ACTIVITYTYPE>
            <DETAILS>TIME</DETAILS>
            <PORTALKEY>15</PORTALKEY>
            <READERKEY>23</READERKEY>
            <PERSONID>527</PERSONID>
            <CARDFORMAT>26 Bit Wiegand FC 7</CARDFORMAT>
            <ENCODEDNUM>555</ENCODEDNUM>
         </PARAMS>
  </COMMAND>
</NETBOX-API>
```

InsertActivity call for user-defined activity text:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="InsertActivity" num="1">
         <PARAMS>
            <ACTIVITYTYPE>USERACTIVITY</ACTIVITYTYPE>
            <ACTIVITYTEXT>The ACMD system is online</ACTIVITYTEXT>
         </PARAMS>
  </COMMAND>
</NETBOX-API>
```

InsertActivity call for Access Granted for a person at an elevator with floor select reporting:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="InsertActivity">
         <PARAMS>
            <ACTIVITYTYPE>ACCESSGRANTED</ACTIVITYTYPE>
            <ELEVATORKEY>6</ELEVATORKEY>
            <FLOORKEY>13</ELEVATORKEY>
            <READERKEY>50</READERKEY>
            <PERSONID>527</PERSONID>
            <CARDFORMAT>26 Bit Wiegand FC 7</CARDFORMAT>
            <ENCODEDNUM>555</ENCODEDNUM>
         </PARAMS>
  </COMMAND>
</NETBOX-API>
```

A successful response to an **InsertActivity** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="InsertActivity" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
```

```
</NETBOX>
```

An error response to an **InsertActivity** call where the specified person ID does not exist:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="InsertActivity" num="1">
       <CODE>FAIL</CODE>
       <DETAILS>
          <ERRMSG>Person not found</ERRMSG>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

ListEvents

The **ListEvents** command retrieves the list of configured Event definitions.

ListEvents request is accepted at address:

```
http://<NetBox IP Address>/nbws/goforms/nbapi
```

The list of events can be used with the **TriggerEvent** command.

See *The Event API* for a detailed discussion of how to trigger an event.

Calling Parameters

None.

Response

This command returns SUCCESS in the CODE element:

- SUCCESS: Returns a DETAILS element block that contains EVENT element blocks, each of which include an event ID, NAME, PARTITIONID, and an ACTIONS block.
- The ID element is necessary because of the additions of paging and NEXTKEY.
- The ACTIONS block includes a set of named actions.
- The NAME and ACTION elements are wrapped as CDATA, Metadata that is not interpreted as XML data.

This command does not return FAIL in the CODE element.

Example

ListEvents call requires that the sessionid both be included in the XML (body) as shown below, and as a cookie in the header of the request:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
  <COMMAND name="ListEvents">
  </COMMAND>
</NETBOX-API>
```

A successful response to the **ListEvents** call returns a list of events and includes CDATA:

```
<NETBOX>
   <RESPONSE command="ListEvents" num="1">
      <CODE>SUCCESS</CODE>
      <DETAILS>
         <EVENTS>
            <EVENT>
                <ID>1</ID>
                <NAME>
                   <![CDATA[158_backDoorForcedEvent]]>
                </NAME>
                <PARTITIONID>1</PARTITIONID>
                <ACTIONS>
                    <ACTION><![CDATA[SentToJohnBates]]></ACTION>
                    <ACTION><![CDATA[Activate Output 2]]></ACTION>
                    <ACTION><![CDATA[Log Event]]></ACTION>
                </ACTIONS>
```

```
            </EVENT>
                 .
                 .
                 .

            <EVENT>
                <NAME><![CDATA[156_frontDoorHeldEvent]]></NAME>
                <PARTITIONID>1</PARTITIONID>
                <ACTIONS>
                    <ACTION><![CDATA[Log Event]]></ACTION>
                    <ACTION><![CDATA[Activate Output]]></ACTION>
                    <ACTION><![CDATA[SendToJohnBates]]></ACTION>
                </ACTIONS>
            </EVENT>
        </EVENTS>
        <NEXTKEY>-1</NEXTKEY>
      </DETAILS>
    </RESPONSE>
</NETBOX>
```

LockPortal

The **LockPortal** command requests that a portal specified by a PORTALKEY be set to the Ready (locked) state.

Use **UnlockPortal** to request that a portal be set to the Extended Unlock state.

Use **MomentaryUnlockPortal** to request that a portal be momentarily unlocked.

Calling Parameters

- PORTALKEY: Key associated with the portal to be set to the Locked state.

  Use **GetOutputs** to retrieve the list of PORTALKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PORTALKEY associated with the portal to be locked in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing PORTALKEY
  - Invalid portal key
  - Portal not online or unreachable
  - Portal state not changed
  - Command not supported for this device type
  - SQL function returned unexpected value

Example

LockPortal call to lock the portal associated with PORTALKEY value of 7:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="LockPortal" num="1">
       <PARAMS>
          <PORTALKEY>7</PORTALKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **LockPortal** command response that indicates that the portal associated with PORTALKEY 7 has been locked:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="LockPortal" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <PORTALKEY>7</PORTALKEY>
       </DETAILS>
    </RESPONSE>
</NETBOX>
```

Login

The **Login** command logs into a partition of a system. The default partition is "Master."

Use **SwitchPartition** to switch the current partition.

Calling Parameters

- USERNAME: Username of a system user with full system setup, partition setup or user role mapping API privileges.
- PASSWORD: Password of a system user with full system setup, partition setup or user role mapping API privileges.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the session ID. Use this session ID in subsequent commands.

  The session ID remains active until the session times out or a **Logout** call is issued.

- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing USERNAME
  - Missing PASSWORD

Example

Login call to log into the "Master" partition:

```
<NETBOX-API>
    <COMMAND name="Login" num="1">
        <PARAMS>
            <USERNAME>jsmith</USERNAME>
             <PASSWORD>jsmith</PASSWORD>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **Login** call that returns the session ID:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="Login" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

The session ID is included in the next command:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetReader" num="1">
        <PARAMS>
           <READERKEY>16</READERKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Logout

The **Logout** command logs out of an API session created with a Login command. Use the session ID to log out of the session.

Calling Parameters

This command has no calling parameters.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: No additional elements are returned.
- `FAIL`: Returns no error messages in the `ERRMSG` element in the `DETAILS` element block.

Example

Logout call to log out of the current session:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
   <COMMAND name="Logout" num="1">
   </COMMAND>
</NETBOX-API>
```

Successful response to the **Logout** call:

```
<NETBOX>
    <RESPONSE command="Logout" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyAccessLevel

The **ModifyAccessLevel** command modifies an existing access level.

Calling Parameters

- ACCESSLEVELKEY: The access level key.
- ACCESSLEVELNAME (optional): The access level name of up to 64 characters.
- ACCESSLEVELDESCRIPTION (optional): The access level description.
- READERKEY (optional): Key corresponding to a reader assigned to this access level.
- READERGROUPKEY (optional): Key corresponding to the reader group assigned to this access level.
- TIMESPECGROUPKEY (required): Key corresponding to the time spec group assigned to this access level.
- THREATLEVELGROUPKEY (optional): Key corresponding to the threat level group assigned to this access level.

Use **GetAccessLevels**, **GetReaders**, **GetReaderGroups**, or **GetTimeSpecGroups** to retrieve a list of ACCESSLEVELKEY, READERKEY, READERGROUPKEY, or TIMESPECGROUPKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing ACCESSLEVELKEY.
  - NOT FOUND.
  - Invalid ACCESSLEVELKEY

Example

ModifyAccessLevel call to assign the reader group, "Reader Group Floors 2/3", to access level,"24/7/365," requires:

- The access level key for the access level, "24/7/365".
- The reader group key for reader group, "Reader Group Floors 2/3".

GetAccessLevels call to retrieve a list of access levels:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="GetAccessLevels" num="1">
       <PARAMS>
          <STARTFROMKEY>0</STARTFROMKEY>
       </PARAMS>
    </COMMAND>
```

```
</NETBOX-API>
```

The **GetAccessLevels** call returns a list of access levels.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevels" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>24/7/365</ACCESSLEVEL>
                <ACCESSLEVEL>Cleaning Crew</ACCESSLEVEL>
                <ACCESSLEVEL>Contractor</ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTNAME></NEXTNAME>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

A second **GetAccessLevels** call:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="GetAccessLevels" num="1">
        <PARAMS>
           <WANTKEY>TRUE</WANTKEY>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

The second **GetAccessLevels** call returns a list of access level keys. The access level key that corresponds to access level "24/7/365" has a value of 1.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetAccessLevels" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <ACCESSLEVELS>
                <ACCESSLEVEL>1</ACCESSLEVEL>
                <ACCESSLEVEL>2</ACCESSLEVEL>
                <ACCESSLEVEL>3</ACCESSLEVEL>
            </ACCESSLEVELS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

The **GetReaders** call returns a list of reader group keys. The reader group key that corresponds to reader group "Reader Group Floors 2/3" has a value of 28.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="GetReaderGroups" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <READERGROUPS>
                <READERGROUP>
                    <READERGROUPKEY>28</READERGROUPKEY>
                    <NAME>Reader Group Floors 2/3</NAME>
                    <DESCRIPTION>Stairwell 2 and 3</DESCRIPTION>
                   <READERS>
                        <READER>
                            <READERKEY>17</READERKEY>
```

```
                            <NAME>Second Floor</NAME>
                        </READER>
                        <READER>
                            <READERKEY>2</READERKEY>
                            <NAME>Third Floor</NAME>
                        </READER>
                    </READERS>
                </READERGROUP>
                <READERGROUP>
                    <READERGROUPKEY>29</READERGROUPKEY>
                    <NAME>Reader Group Floor 2</NAME>
                    <DESCRIPTION>Stairwell 2 Only</DESCRIPTION>
                    <READERS>
                        <READER>
                            <READERKEY>17</READERKEY>
                            <NAME>Second Floor</NAME>
                        </READER>
                    </READERS>
                </READERGROUP>
                <READERGROUP>
                    <READERGROUPKEY>30</READERGROUPKEY>
                    <NAME>Reader Group Floor 3</NAME>
                    <DESCRIPTION>Stairwell 3 only</DESCRIPTION>
                    <READERS>
                        <READER>
                            <READERKEY>2</READERKEY>
                            <NAME>Third Floor</NAME>
                        </READER>
                    </READERS>
                </READERGROUP>
              </READERGROUPS>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

ModifyAccessLevel call to request that the reader group corresponding to `READERGROUPKEY` 28 is assigned to the access level corresponding to the `ACCESSLEVELKEY` with a value of 1:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="ModifyAccessLevel" num="1">
         <PARAMS>
              <ACCESSLEVELKEY>1</ACCESSLEVELKEY>
              <READERGROUPKEY>28</READERGROUPKEY>
          </PARAMS>
        </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyAccessLevel** call.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyAccessLevel" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyAccessLevelGroup

The **ModifyAccessLevelGroup** command modifies an existing access level group.

Specifying the group name, description and/or list of new access levels are optional and replace the values.

You cannot change a single-partition access level group to a multi-partition access level group.

Adding access levels from other partitions to a single-partition access level group will result in an error.

Calling Parameters

- `ACCESSLEVELGROUPKEY`: The access level groupkey.

  Use **GetAccessLevels** to retrieve the list of `ACCESSLEVELGROUPKEY` values.

- `NAME` (optional): The access level group name of up to 64 characters.
- `DESCRIPTION` (optional): The access level group description.
- `ACCESSLEVELS` (optional): List of access levels to be included in the group. Each `ACCESSLEVELS` element block includes:
  - `PARTITIONKEY` (required for multi-partition access level groups where `NAME` is specified): Partition of the access level.

    Use **GetAccessLevelGroups** to retrieve the list of access levels.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns the `ACCESSLEVELGROUPKEY` in the `DETAIL` element block as the identifier for the new access level group.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Access Level Group Name exceeds max size of 64 characters
  - Access Level [Name= XXX}] does not exist in Access Level Group Partition X
  - Cannot combine KEY with NAME or PARTITIONKEY params
  - Unable to add or modify access level group: invalid access level parameter(s)
  - Partition of one or more access levels do not match access level group partition
  - Cannot change a single partitioned group to a multi-partitioned or vice versa.
  - PARTITIONKEY of an access level is a required field when the access level name is specified when group is cross partition.

Example

ModifyAccessLevelGroup call to request that the access level group corresponding to `ACCESSLEVELGROUPKEY` 75 has a new name and description, and be assigned the access level "Main Doors":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyAccessLevelGroup" num="1">
        <PARAMS>
            <ACCESSLEVELGROUPKEY>75</ACCESSLEVELGROUPKEY>
            <NAME>All Developers</NAME>
            <DESCRIPTION>All Developer Access</DESCRIPTION>
            <ACCESSLEVELS>
                <ACCESSLEVEL>
                    <NAME>Main Doors</NAME>
                </ACCESSLEVEL>
            </ACCESSLEVELS>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyAccessLevelGroup** call.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyAccessLevelGroup" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyCredential

The **ModifyCredential** command modifies a credential currently assigned to a person.

Calling Parameters

- PERSONID: ID number of the person to whom the credential is assigned.
- CARDFORMAT: Name of the format to be used to identify the credential.

  This parameter cannot be used to modify an existing credential.
- ENCODEDNUM: The encoded number for the credential.

  This parameter cannot be used to modify an existing credential.

  The value for ENCODEDNUM must be an integer that fits within the number of bits specified in the CARDFORMAT for that credential.
- HOTSTAMP (optional): The hotstamp number for the credential.

  This parameter can be used to modify an existing credential.
- CREDENTIALID: USED WITH PERSONID TO SPECIFY THE CREDENTIAL TO BE MODIFIED AND THE PERSON TO WHOM IT IS ASSIGNED—AS AN ALTERNATIVE TO SPECIFYING THE ENCODED NUMBER AND CREDENTIAL FORMAT.

  IF CREDENTIALID IS INCLUDED IN A CALL, ONLY THE CREDENTIAL'S ATTRIBUTES CAN BE MODIFIED; ITS FORMAT, ENCODED NUMBER, AND HOTSTAMP NUMBER CANNOT BE MODIFIED.

Note: The credential ID is an alias for the actual credential number. As a security measure, credential IDs can be retrieved and stored in a client system in place of the encoded numbers and/or hot stamps. This will allow people to manage credentials from the client system without seeing the actual credential numbers

USE **AddCredential** TO RETRIEVE CREDENTIAL IDS FOR CREDENTIALS WHEN THEY ARE ADDED. USE **GetPerson** AND **SearchPersonData** TO RETRIEVE CREDENTIAL IDS FOR EXISTING CREDENTIALS.

- DISABLED (optional): Use 1 to disable the credential or 0 to enable the credential.

  This parameter cannot be included in a call that includes the CARDSTATUS parameter.
- CARDSTATUS (optional): Text string that specifies the status of the credential.

  This parameter cannot be included in a call that includes the DISABLED parameter.

Note: An administrator can use the *Credential Attributes* page (under *Configuration : Access Control* in the NetBox web interface) to customize the names of credential status settings.

- CARDEXPDATE: Expiration date for the credential. See *Date Formats*.

  A valid Credential Expiration date must be passed as a parameter if *Enable Credential Expiration Requirement* is selected on the Access Control tab of the Network Controller page.

NOTE: CARDSTATUS AND CARDEXPDATE ARE NOT SUPPORTED FOR THE GLOBAL API.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates the credential has been successfully modified. No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Invalid PersonID
  - Credential expiration date is required

Example

ModifyCredential call to change credential's status to Active, using CREDENTIALID with PERSONID to identify the credential to be modified.

```
<NETBOX-API sessionid="900493536">
     <COMMAND name="ModifyCredential" num="1">
         <PARAMS>
           <PERSONID>_5</PERSONID>
           <CARDSTATUS>Active</CARDSTATUS>
           <CREDENTIALID>4</CREDENTIALID>
         </PARAMS>
    </COMMAND>
</NETBOX-API>
```

ModifyCredential call to disable a credential, using CREDENTIALID with PERSONID to identify the credential to be modified.

```
<NETBOX-API sessionid="900493537">
     <COMMAND name="ModifyCredential" num="1">
         <PARAMS>
           <PERSONID>_5</PERSONID>
           <DISABLED>1</DISABLED>
           <CREDENTIALID>5</CREDENTIALID>
         </PARAMS>
    </COMMAND>
</NETBOX-API>
```

ModifyCredential call to modify a credential's expiration date, using CREDENTIALID with PERSONID to identify the credential to be modified.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="ModifyCredential" num="1">
         <PARAMS>
           <PERSONID>_5</PERSONID>
```

```
            <CARDEXPDATE>2018-10-03</CARDEXPDATE>
            <CREDENTIALID>6</CREDENTIALID>
          </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to each of the **ModifyCredential** calls:

```
<NETBOX sessionid="90049353n">
    <RESPONSE command="ModifyCredential" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyHoliday

The **ModifyHoliday** command modifies an existing holiday.

Calling Parameters

- HOLIDAYKEY: A holiday key.

  Use **GetHoliday**s to retrieve the list of HOLIDAY key values.

- HOLIDAYNAME: Name of the holiday.

  Use **GetHoliday** to retrieve the list of HOLIDAYNAME values.

- HOLIDAYGROUPS: Specify any or all of the holiday group values (1, 2, 3, 4, 5, 6, 7, 8 separated by commas).
- STARTDATE: Start date in YYYY:MM:DD HH:MM format. See *Date Formats*.
- ENDDATE: End Time in YYYY:MM:DD HH:MM format. See *Date Formats*.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates the holiday was successfully modified. No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing HolidayKey
  - NOT FOUND.
  - Invalid HolidayKey.
  - STARTDATE and ENDDATE must be specified in YYYY-MM-DD HH:MM or YYYY-MM-DD HH:MM:SS format.

Example

ModifyHoliday call to modify the holiday definition for "Thanksgiving" in "Holiday group 1":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyHoliday" num="1">
     <PARAMS>
           <HOLIDAYKEY>1</HOLIDAYKEY>
           <HOLIDAYNAME>Thanksgiving</HOLIDAYNAME>
           <HOLIDAYGROUPS>1</HOLIDAYGROUPS>
           <STARTDATE>2016-12-25 00:00:00<STARTDATE>
           <ENDDATE>2016-12-26 00:00:00</ENDDATE>
       </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyHoliday** call indicates that the holiday definition has been modified.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
```

```
<RESPONSE command="ModifyHoliday" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyPerson

The **ModifyPerson** command allows you to modify an existing person record. The PERSONID parameter is required, and the command will fail if it does not match the person ID of an existing person.

You can use ModifyPerson to delete or undelete a person record. If you have the full system setup user role, or a custom role that gives you the Purge Person permission, you can also use the command to purge person records.

Calling Parameters

- PERSONID – ID of the person for whom the credential is to be modified.
- FIRSTNAME: First name of person whose record is being modified.
- MIDDLENAME: Middle name or middle initial of person whose record is being modified.
- LASTNAME: Last name of person whose record is being modified.
- USERNAME: Username associated with a person's NetBox login account. This is a required field.
- PASSWORD: Password associated with a person's NetBox login account.

  If the AUTHTYPE is SSO or LDAP, then PASSWORD is not applicable and should not be set. If AUTHTYPE is DB, then PASSWORD is required.
- ROLE: Role for a person's record when they Login to the NetBox. This is a required field.
- AUTHTYPE: Authorization type for a person's record. Valid values are DB, LDAP, or SSO. This is a required field.
- MOBILEPHONE: Mobile phone number.
- MSUENABLED: Enables Mobile Security User (MSU) mobile credentials. Email address (CONTACTEMAIL) or mobile number (MOBILEPHONE) are required to enable MSU.
- BLUEDIAMONDENABLED: Enables BlueDiamond mobile credentials.
- NOTES: Notes field of person record.
- EXPDATE: Expiration date for person record in YY-MM-DD format. See *Date Formats*.
- ACTDATE: Activation date for person record in YY-MM-DD format. See *Date Formats*.
- UDF1...UDF20: User-defined fields (20).
- PIN: PIN number that may be required for a card reader with a numeric pad.
- PERSONPURGE: If TRUE, removes the person record and all data related to it from the NetBox system permanently. Because person records are purged via a batch process, it will take some time for all data related to the person record to be removed from the system.

Notes: Purging a person record will not remove related information from archive files—whether they are stored locally on the controller or in an off-controller location, such as an FTP server or network share. Unless you remove this information manually, it could continue to appear in the system; for example, in reports that are configured to search archives.

If a person record is purged from a NetBox that is connected to a Global system, Global will send the purge request to all other connected NetBox systems that are running software version 5.0 or later—and these 5.0 or later systems will also purge the record. If a connected NetBox is running an earlier software version, Global will purge the record from that system when it is upgraded to software version 5.0 or later.

You cannot use the API to purge person records from Global. The Global purge capability is available only via Data Operations.

- `ACCESSLEVELS`: Element block containing one or more access levels (maximum of 32) to be associated with the person. Access levels in excess of the maximum will be silently ignored. For an access level to be added to a person record, it must already have been defined in your NetBox. Otherwise, the command will fail. The `ACCESSLEVELS` element block contains a separate `ACCESSLEVEL` element for each access level.

  There are two forms of syntax: Access Levels by Name only and Access Levels by Name and Flag. See Person Access Levels for a detailed explanation.

  The first form of syntax is "Access Levels by Name only". The `ACCESSLEVELS` element block contains a single element for each `ACCESSLEVEL`.

  - `ACCESSLEVEL`: Access level name.

    If one or more access levels are passed with the **ModifyPerson** command, the API will first clear all existing access levels from the Person record before adding new ones. Thus, all access levels that should continue to be associated with a Person record must be specified in a call to **ModifyPerson**.

  The second form of syntax is "Access Levels by Name and flag" and may be referred to as "Alternative Syntax". The `ACCESSLEVELS` element block contains multiple elements for each `ACCESSLEVEL`:

  - `ACCESSLEVEL`: Access level element block
  - `ACCESSLEVELNAME`: Access level name.
  - `DELETE`: A value of 1 for DELETE will result in an access level being removed from the person record. A value of 0 will result in an access level being added to the person record.
  - `ACTDATE`: NO VALUE MEANS NOW. A DATE STRING MEANS SET THE DATE TO THE VALUE PASSED IN. EXCLUSION OF THE PARAMETER MEANS LEAVE THE EXISTING VALUE. THIS VALUE IS NOT ENFORCED, ONLY RECORDED.

  - EXPDATE: NO VALUE MEANS NEVER. A DATE STRING MEANS SET DATE TO THE VALUE PASSED IN. EXCLUSION OF THE PARAMETER MEANS LEAVE THE EXISTING DATE.
  - AUTOREMOVE: A VALUE OF 1 MEANS THE ACCESS LEVEL WILL BE REMOVED FROM THE PERSON RECORD WHEN THE ACCESS LEVEL EXPIRES.

    Any access level already associated with a person record, but not specified in a **ModifyPerson** call, will continue to be associated with that person record.

- PICTURE: (optional) Picture data for the person being added. This data must be Base64 encoded and not exceed 650KB in size. (Note that the maximum size returned in the GetPicture command is 120KB.)
- PICTUREEXT: Extension that describes the format of the picture data (for example, "jpg"). This parameter is required if PICTURE is supplied.
- PICTUREURL: (optional): Name of file for picture data as it will be stored on the system. If not specified, the API will assign a filename with the format:

  lastname firstname.extension
- BADGELAYOUT: Name of the photo ID badging layout file.
- CONTACTPHONE: Office phone number.
- CONTACTEMAIL: Office email address.
- CONTACTLOCATION: Office location.
- OTHERCONTACTNAME: Emergency contact name.
- OTHERCONTACTPHONE1: Emergency contact phone number.
- OTHERCONTACTPHONE2: Alternate emergency contact phone number.
- PRINTBADGE: Add a badge print request to, or remove a badge print request from, the person record. A value of TRUE adds a badge print request. A value of FALSE removes a badge print request.
- VEHICLES: The VEHICLES element block contains a set of VEHICLE elements for each vehicle:
  - VEHICLECOLOR
  - VEHICLEMAKE
  - VEHICLEMODEL
  - VEHICLESTATE
  - VEHICLELICNUM
  - VEHICLETAGNUM
  - DELETE: Specify a value of 1 for DELETE to remove the vehicle from the person record. Specify a value of 0 (or no value) to keep the vehicle.

If you use this tag to delete a vehicle, include sufficient information to ensure that the correct vehicle will be removed.

- `DELETED`: If `TRUE`, removes the person record from the database and has the same result as a **RemovePerson** call. Supplying `FALSE` to a person record that has been deleted, undeletes the person record.
- `ALLPARTITIONS`: If `TRUE`, enables you to modify a person record in a different partition. If `FALSE` or no parameter is specified, the person record must be in the current partition.

  The `ALLPARTITIONS` parameter requires *User Login Authentication* and the account that the API client uses to log in must have full system setup privilege.
- `PARTITIONKEY` (Global API Only): The partition to which the **ModifyPerson** call applies to when the application has not set a partition with **SwitchPartition**.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Indicates that the person record has been successfully updated.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Person ID Required is checked, cannot update to an empty value.
  - Missing PERSONID
  - Missing LASTNAME
  - The New Person ID exists in the system already and Enforce Unique is checked, please change it.
  - Vehicle Delete Tag has to be a 1 or 0.
  - Access level does not exist: XX
  - Badging is not Enabled or Configured in the System. Cannot set Badge Print Request.
  - ACTDATE and EXPDATE must follow YYYY-MM-DD hh:mm[:ss] or YYYY-MM-DD format

Example

ModifyPerson call to add a middle initial to the person record with ID number 1001:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyPerson" num="1">
        <PARAMS>
           <PERSONID>1001</PERSONID>
           <MIDDLENAME>V</MIDDLENAME>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyPerson** call adds the person record:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyPerson" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PERSONID>1001</PERSONID>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

A more complex command using multiple **ModifyPerson** parameters:

Note: The `ACCESSLEVELS` element block contains the `DELETE` element with a value of 0, indicating that the access level is to be added.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyPerson" num="1">
        <PARAMS>
           <PERSONID>1001</PERSONID>
           <NOTES>Reminder: expects purchase new car</NOTES>
          <EXPDATE>2024-01-01</EXPDATE>
           <ACTDATE>2016-10-16</ACTDATE>
           <ACCESSLEVELS>
              <ACCESSLEVEL>24/7/365</ACCESSLEVEL>
              <DELETE>0</DELETE>
           </ACCESSEVELS>
           <UDF1>Analyst</UDF1>
           <UDF2>Mary Bixby</UDF2>
           <UDF3>B6/305</UDF3>
           <UDF4>Framingham MA</UDF4>
           <PIN>1111</PIN>
           <CONTACTPHONE>999-123-4567</CONTACTPHONE>
           <CONTACTEMAIL>jappleby@yahoo.com</CONTACTEMAIL>
           <CONTACTSMSEMAIL>jappleby@gmail.com</CONTACTSMSEMAIL>
           <CONTACTLOCATION>Building 2</CONTACTLOCATION>
           <OTHERCONTACTNAME>Mary Appleby</OTHERCONTACTNAME>
           <OTHERCONTACTPHONE1>999-123-4568</OTHERCONTACTPHONE1>
           <OTHERCONTACTPHONE2>999-123-4569</OTHERCONTACTPHONE2>
           <PRINTBADGE>TRUE</PRINTBADGE>
           <VEHICLES>
```

```
            <VEHICLE>
               <VEHICLECOLOR>Blue</VEHICLECOLOR>
               <VEHICLEMAKE>Honda</VEHICLEMAKE>
               <VEHICLEMODEL>Honda CRV</VEHICLEMODEL>
               <VEHICLESTATE>MA</VEHICLESTATE>
               <VEHICLELICNUM>123 PGA</VEHICLELICNUM>
               <VEHICLETAGNUM>TAG 3000</VEHICLETAGNUM>
            </VEHICLE>
          <VEHICLE>
               <VEHICLECOLOR>Red</VEHICLECOLOR>
               <VEHICLEMAKE>Buggatti</VEHICLEMAKE>
               <VEHICLEMODEL>XV32</VEHICLEMODEL>
               <VEHICLESTATE>MA</VEHICLESTATE>
               <VEHICLELICNUM>456 PGA</VEHICLELICNUM>
               <VEHICLETAGNUM>TAG 3001</VEHICLETAGNUM>
            </VEHICLE>
         </VEHICLES>
      </PARAMS>
   </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyPerson** call adds the person record.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyPerson" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PERSONID>1001</PERSONID>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

ModifyPerson call to purge the person record with ID number 1001:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyPerson" num="1">
       <PARAMS>
          <PERSONID>1001</PERSONID>
          <PERSONPURGE>TRUE
          </PERSONPURGE>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyPerson** call indicates that the person record has been purged:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyPerson" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyPortalGroup

The **ModifyPortalGroup** command modifies an existing portal group.

Calling Parameters

- PORTALGROUPKEY: The portal group key.

  Use **GetPortalGroups** to retrieve the list of PORTALKEY, PORTALGROUPKEY, TIMESPECGROUPKEY, THREATLEVELGROUPKEY and UNLOCKTIMESPECGROUPKEY values.

- NAME (optional): Portal group name of up to 64 characters.
- DESCRIPTION (optional): Portal group name description.
- PORTALKEY (required): Key corresponding to a portal assigned to this portal group.
- UNLOCKTIMESPECGROUPKEY (optional): Key corresponding to the time spec group for unlocking portals in this group.
- THREATLEVELGROUPKEY (optional): Key corresponding to the threat level group for portals in this group.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - PORTAL KEY XXX DOES NOT EXIST
  - Missing PORTALGROUPKEY
  - Portal Group Name exceeds is 64 characters
  - Not Found – indicates portal group key does not exist.
  - Invalid PORTALGROUPKEY

Example

ModifyPortalGroup call to change the description for the portal group corresponding to PORTALGROUPKEY with a value of 56, to "Readers 1, 2, and 3":

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyPortalGroup" num="1">
    <PARAMS>
          <PORTALGROUPKEY>56</PORTALGROUPKEY>
          <PORTALKEY>1</PORTALKEY>
          <PORTALKEY>2</PORTALKEY>
          <DESCRIPTION>Readers 1, 2, and 3</DESCRIPTION>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response indicates that the portal group description was modified:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyPortalGroup" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyReaderGroup

The **ModifyReaderGroup** command modifies an existing reader group.

Calling Parameters

- READERGROUPKEY: The reader group key.

  Use **GetReaderGroups** to retrieve the list of READERGROUPKEY and READERKEYS values.

- NAME (optional): Reader group name of up to 64 characters.
- DESCRIPTION (optional): Reader group name description.
- READERKEYS (optional): Key corresponding to a reader assigned to this reader group.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns no additional elements.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing READERGROUPKEY
  - Not Found
  - A field value with a maximum of 64 characters was too long.
  - Invalid READERGROUPKEY.

Example

ModifyReaderGroup call to change the description for the reader group corresponding to the READERGROUPKEY with a value of 28, to "Stairwells Floors 1 and 2":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyReaderGroup" num="1">
     <PARAMS>
           <READERGROUPKEY>28</READERGROUPKEY>
           <DESCRIPTION>Stairwells Floors 1 and 2</DESCRIPTION>
       </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyReaderGroup** call response indicates that the reader group description was modified:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyReaderGroup" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyThreatLevel

The **ModifyThreatLevel** command modifies a threat level.

Calling Parameters

- `LEVELNAME`: Name of threat level.
- `SEQNUM`: Display order for the threat level in the user interface.
- `COLOR`: Specify White, Green, Blue, Yellow, Orange, or Red as desired.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns no additional elements.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Not Found

Example

ModifyThreatLevel call to request to modify the threat level, "Intruder Alert", so that the display order is 4 and the color is "Blue":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="ModifyThreatLevel" num="1">
        <PARAMS>
           <LEVELNAME>Intruder Alert</LEVELNAME>
           <SEQNUM>4</SEQNUM>
           <COLOR>Blue<COLOR>
        </PARAMS>
</COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyThreatLevel** call that indicates the threat level was modified.

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="ModifyThreatLevel" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

ModifyThreatLevelGroup

The **ModifyThreatLevelGroup** command modifies a threat level group.

Calling Parameters

- LEVELGROUPNAME: Name of threat level group.
- LEVELNAMES: Element block containing LEVELNAME elements.
- LEVELNAME: LEVELNAME element block for each threat level to be defined in the threat level group.

  The **ModifyThreatLevelGroup** call replaces all members of the threat level group with the new threat level name(s) in the LEVELNAME element block.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns no additional elements.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Not Found

Example

ModifyThreatLevelGroup call to request that the threat level, "Minor", is added to the threat level group. "Intruder Alert":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="ModifyThreatLevelGroup" num="1">
        <PARAMS>
           <LEVELGROUPNAME>Intruder Alert</LEVELGROUPNAME>
           <LEVELNAMES>
             <LEVELNAME>Default</LEVELNAME>
             <LEVELNAME>Minor</LEVELNAME>
             <LEVELNAME>Guarded</LEVELNAME>
             <LEVELNAME>Elevated</LEVELNAME>
             <LEVELNAME>High</LEVELNAME>
             <LEVELNAME>Severe</LEVELNAME>
           </LEVELNAMES>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyThreatLevelGroup** call indicates that the threat level group was modified to include the threat level names specified in the command:

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="ModifyThreatLevelGroup" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

ModifyTimeSpec

The **ModifyTimeSpec** command modifies a time spec.

Calling Parameters

- TIMESPECKEY: A time spec key.

  Use **GetTimeSpecs** to retrieve the list of TIMESPECKEY values.

- NAME: Name of the time spec.
- DESCRIPTION: Description of the time spec.
- STARTTIME: Start Time in HH:MM format.
- ENDTIME: End Time in HH:MM format.
- Days of the week to be included, 1 for TRUE and 0 for FALSE, for each of:

  MONDAY

  TUESDAY

  WEDNESDAY

  THURSDAY

  FRIDAY

  SATURDAY

  SUNDAY

- HOLIDAYGROUPS: Specify any of all of the holiday group numbers (1,2,3,4,5,6,7,8 separated by commas).

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns no additional elements.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - 'Always' cannot be changed.
  - 'Never' cannot be changed.
  - Missing TIMESPECKEY
  - INVALID TIMESPECKEY
  - NOT FOUND
  - BAD VALUE FOR WEEK DAY: MUST BE 1 OR 0
  - Time Spec name already exists: <Time spec name>
  - Time Spec name cannot be null.

- Time Spec name cannot be empty.

Example

ModifyTimeSpec call to modify the description, start time, and end time of the time spec corresponding to the `TIMESPECKEY` with a value of 12:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyTimeSpec" num="1">
     <PARAMS>
           <TIMESPECKEY>12</TIMESPECKEY>
           <DESCRIPTION>Part-time 9 to 1</DESCRIPTION>
           <STARTTIME>09:00</STARTTIME>
           <ENDTIME>01:00></ENDTIME>
       </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyTimeSpec** call indicates that the time spec description has been updated:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
<RESPONSE command="ModifyTimeSpec" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

ModifyTimeSpecGroup

The **ModifyTimeSpecGroup** command modifies a time spec group.

Calling Parameters

- TIMESPECGROUPKEY: Element containing time spec group key.
- NAME: Name of the time spec group (optional).
- DESCRIPTION: Description of the time spec group (optional).
- TIMESPECKEYS: Element block containing the TIMESPECKEY elements.

The **ModifyTimeSpecGroup** call replaces all members of the time spec group.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns no additional elements.
- FAIL: RETURNS AN ERROR MESSAGE IN the ERRMSG element in the DETAILS element block:
  - Missing TIMESPECGROUPKEY
  - Invalid TIMESPECGROUPKEY
  - NOT FOUND

Example

This is a **ModifyTimeSpecGroup** call to request that the threat level group corresponding to the TIMESPECGROUPKEY with a value of 65, is modified to contain only the time specs corresponding to TIMESPECKEYs with the values of 15, 16, and 17.

Any other time specs that were defined as part of the time spec group before the command, will no longer be assigned to the time spec group.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="ModifyTimeSpecGroup" num="1">
    <PARAMS>
          <TIMESPECGROUPKEY>65</TIMESPECGROUPKEY>
          <TIMESPECKEYS>
          <TIMESPECKEY>15</TIMESPECKEY>
          <TIMESPECKEY>16</TIMESPECKEY>
          <TIMESPECKEY>17</TIMESPECKEY>
          </TIMESPECKEYS>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyTimeSpecGroup** call indicates that the time spec group was modified:

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="ModifyTimeSpecGroup" num="1">
       <CODE>SUCCESS</CODE>
```

```
    </RESPONSE>
</NETBOX>
```

ModifyUDFListItems

The **ModifyUDFListItems** command adds, modifies, and deletes values in an existing UDF (user-defined field) value list.

A maximum of 10 values can be added, modified, or deleted in a call to **ModifyUDFListItems**.

Use **GetUDFLists** to retrieve a list of UDF value lists configured in the system. Use **GetUDFListItems** to retrieve a list of the values in a UDF value list.

Calling Parameters

- `UDFLISTKEY`: The unique key for the UDF value list containing the value to be added, modified, or deleted.
- `ITEMKEY`: The unique key for the UDF list value to be modified or deleted. This key is not used when adding UDF list values.
- `ITEMNAME`: The name of the UDF list value to be added, modified, or deleted.
- `DELETE`: Number specifying whether the UDF list value should be added or modified, or deleted. The number 0 indicates the value should be added or modified. The number 1 indicates the value should be deleted.
- `CUSTOMKEY`: (optional) A unique key to be assigned to the UDF list value being added. Once assigned, the key specifies the value to be modified in the list.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Indicates the UDF list value was successfully added, modified, or deleted. No additional elements are returned.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Missing UDFLISTKEY
  - INVALID UDFLISTKEY
  - NOT FOUND
  - ITEMKEY or CUSTOMKEY is required
  - Item [Item Name: *name*, Item Key: *key*, Custom Key: *customekey*] Could not be Found.
  - Error Updating UDF List Item: [*name*]
  - Error Adding UDF List Item. Name is Required

Example

ModifyUDFListItems calls to do the following:

- `DELETE` a list value using `ITEMKEY 1` from `UDFLISTKEY 1`. Optionally, if the `CUSTOMKEY` for this value is unique, it can be used in place of the `ITEMKEY`.

- `MODIFY` a list value using `ITEMKEY 2` or `CUSTOMKEY_2`. If modifying the `CUSTOMKEY`, the `ITEMKEY` is required.
- `ADD` a list value, using `ITEMNAME`, which is required, and setting an optional `CUSTOMKEY`.

```
<NETBOX-API sessionid="3769758827">
     <COMMAND name="ModifyUDFListItems" num="1" dateformat="tzoffset">
       <PARAMS>
         <UDFLISTKEY>1</UDFLISTKEY>
          <LISTITEMS>
           <LISTITEM>
                    <ITEMKEY>1</ITEMKEY>
                     <DELETE>1</DELETE>
                </LISTITEM>
                <LISTITEM>
                     <ITEMKEY>2</ITEMKEY>
                     <DELETE>0</DELETE>
                     <ITEMNAME>Engineering Manager</ITEMNAME>
                     <CUSTOMKEY>_2</CUSTOMKEY>
                </LISTITEM>
                <LISTITEM>
                     <DELETE>0</DELETE>
                     <ITEMNAME>Marketing Manager</ITEMNAME>
                     <CUSTOMKEY>_3</CUSTOMKEY>
                </LISTITEM>
          </LISTITEMS>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **ModifyUDFListItems** call that indicates all three commands were completed.

```
<NETBOX sessionid="3769758827">
    <RESPONSE command="ModifyUDFListItems" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

MomentaryUnlockPortal

The **MomentaryUnlockPortal** command requests that a portal specified by a PORTALKEY be momentarily unlocked.

Note: The duration is not specified in the **MomentaryUnlockPortal** command, but is included in the definition on the system.

Use **LockPortal** to request that a portal be set to the Ready (locked) state.

Use **UnlockPortal** to request that a portal be set to the Extended Unlock state.

Calling Parameters

- PORTALKEY: Key associated with the portal to be locked.

  Use **GetOutputs** to retrieve the list of PORTALKEY values.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PORTALKEY associated with the portal to be momentarily locked in the DETAILS element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Invalid portal key – the key provided does not match a portal.
  - Portal not online or unreachable – the resource associated with the portal is unavailable.
  - Portal state not changed – the portal was already in the specified state.
  - Command is not supported for this device type.
  - SQL function returned unexpected value.

Example

MomentaryUnlockPortal call to momentarily unlock the portal associated with a PORTALKEY with a value of 7:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="MomentaryUnlockPortal" num="1">
       <PARAMS>
          <PORTALKEY>7</PORTALKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **MomentaryUnlockPortal** command response indicates that the portal associated with a PORTALKEY with the value of 7 has been unlocked:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="MomentaryUnlockPortal" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
```

```
            <PORTALKEY>7</PORTALKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

PingApp

The **PingApp** command sends a ping to the system to verify that the network connection is active.

Calling Parameters

The **PingApp** command has no calling parameters.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns no additional elements.
- `FAIL`: Returns no error messages in the `ERRMSG` element in the `DETAILS` element block.

Example

PingApp call to send a ping to the system:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
   <COMMAND name="PingApp" num="1">
   </COMMAND>
</NETBOX-API>
```

Successful response to the **PingApp** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="PingApp" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

RemoveCredential

The **RemoveCredential** command removes a credential that is currently assigned to a person in the system database.

Calling Parameters

- PERSONID: ID number of the person to whom the credential is assigned.

  CARDFORMAT: Name of the format to be used to identify the credential.

- Either or both of the following elements must be included in any call that does not include the CREDENTIALID parameter:
  - ENCODEDNUM : The encoded number for the credential. If this parameter is not supplied, the HOTSTAMP value is assigned to the ENCODEDNUM parameter.
  - HOTSTAMP : The hotstamp number for the credential. If this parameter is not supplied, the ENCODEDNUM value is assigned to the HOTSTAMP parameter.
- CREDENTIALID: USED WITH PERSONID TO SPECIFY THE CREDENTIAL TO BE REMOVED AND THE PERSON TO WHOM IT IS ASSIGNED.

Note: The credential ID is an alias for the actual credential number. As a security measure, credential IDs can be retrieved and stored in a client system in place of the encoded numbers and/or hot stamps. This will allow people to manage credentials from the client system without seeing the actual credential numbers.

SEE **AddCredential** FOR INFORMATION ON RETRIEVING CREDENTIAL IDS FOR CREDENTIALS WHEN THEY ARE ADDED. SEE **GetPerson** AND **SearchPersonData** FOR INFORMATION ON RETRIEVING CREDENTIAL IDS FOR EXISTING CREDENTIALS.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Indicates the credential was successfully removed. No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Missing required param, must supply Person ID and Credential ID
  - Person NOT FOUND
  - INVALID PERSONID
  - CARDFORMAT NOT FOUND
  - NOT FOUND
  - Either the Credential ID is invalid, or it does not belong to Person ID

- Missing required parameter. Must supply either Encoded Number or Hotstamp with Card Format, or Credential ID

Example

RemoveCredential call to remove a credential associated with Person ID 1005:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="RemoveCredential" num="1">
       <PARAMS>
          <PERSONID>1005</PERSONID>
          <CREDENTIALID>5</<CREDENTIALID>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **RemoveCredential** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="RemoveCredential" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

RemovePerson

The **RemovePerson** command removes a person and all associated credentials from the system database.

Calling Parameters

- `PERSONID`: Person ID for the person whose person record is removed.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: No additional elements are returned.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Person NOT FOUND
  - Missing PERSONID

Example

RemovePerson call to remove the person record associated with Person ID 1004:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="RemovePerson" num="1">
        <PARAMS>
           <PERSONID>1004</PERSONID>
        </PARAMS>
     </COMMAND>
</NETBOX-API>
```

Successful response to the **RemovePerson** call:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="RemovePerson" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

RemoveThreatLevel

The **RemoveThreatLevel** command removes a threat level.

Calling Parameters

- `LEVELNAME`: Threat level name.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: No additional elements are returned.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Not Found

Example

RemoveThreatLevel call to remove a threat level:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="RemoveThreatLevel" num="1">
       <PARAMS>
          <LEVELNAME>Minor</LEVELNAME>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **RemoveThreatLevel** call indicates that the threat level was removed:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="RemoveThreatLevel" num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

RemoveThreatLevelGroup

The **RemoveThreatLevelGroup** command removes a threat level group.

Calling Parameters

- LEVELGROUPNAME: Name of threat level group.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: No additional elements are returned.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Not Found

Example

RemoveThreatLevelGroup call to remove the threat level group, "Intruder Alert":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="RemoveThreatLevelGroup" num="1">
       <PARAMS>
          <LEVELGROUPNAME>Intruder Alert</LEVELGROUPNAME>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **RemoveThreatLevelGroup** call indicates that the threat level group was removed:

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="RemoveThreatLevelGroup" num="1">
       <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

RemoveVirtualCredentialRequest

The **RemoveVirtualCredentialRequest** command removes a mobile credential assignment from a person.

Calling Parameters

- PERSONID: ID of the person associated with the credential to be removed.
- CARDFORMAT: Name of the format to be used to decode the credential.

Note: PERSONID and CARDFORMAT are mandatory parameters.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the CARDFORMAT , PERSONID, STATUS of the removed virtual credential in the DETAIL element block.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - CARDFORMAT NOT FOUND
  - Cannot delete Virtual Credential which is in an x state. Examples of states include Delete Pending, Not Supported.

Example

RemoveVirtualCredentialRequest call to request remove mobile credential with a specific CARDFORMAT that is assigned to a person with PERSONID:

```
<NETBOX-API>
    <COMMAND name="RemoveVirtualCredentialRequest" num="1">
        <PARAMS>
            <PERSONID>123</PERSONID>
            <CARDFORMAT>LenelS2 48bit Corp 1000</CARDFORMAT>
        </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to RemoveVirtualCredentialRequest call:

```
<NETBOX>
    <RESPONSE command=" RemoveVirtualCredentialRequest " num="1">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

SearchPersonData

The **SearchPersonData** command retrieves information from one or more existing person records, based on various search criteria.

Use **GetPerson** to retrieve information from a specific person record.

When calling the **SearchPersonData** command, there are various options in making the call:

- If `PERSONID` is supplied in the call, exactly one matching person record is returned.
- If `PERSONID` is *not* supplied in the call, all matching person records are retrieved.
- If no parameters are supplied in the call, all person records are retrieved.

Data is returned for as many matching person records as will fit in the internal buffer space. If there is insufficient space in the internal buffer, `NEXTKEY` is returned as the value used for the `STARTFROMKEY` parameter in the next **SearchPersonData** call.

When there is no more data to retrieve, `NEXTKEY` is returned with a value of -1.

Calling Parameters

- `PERSONID`: Specifies the starting `PERSONID`. This is used in conjunction with `NEXTKEY` to retrieve the next set of person records.
  - `NEXTKEY` is returned with a value of -1, if there are no more people to be returned.
  - If `NEXTKEY` is returned with a value greater than 0, a `PERSONID` parameter value can be passed in the next **SearchPersonData** command call to return the next set of person records.
- `LASTNAME`: Last name.
- `FIRSTNAME`: First name.
- `MIDDLENAME`: Middle initial.
- `CONTACTEMAIL`: Office email address.
- `MOBILEPHONE`: Mobile phone number.
- `CARDFORMAT`: Card format name.
- `CARDSTATUS`: Card status name.
- `MSUENABLED`: MSU mobile credentials enabled (boolean)
- `BLUEDIAMONDENABLED`: BlueDiamond mobile credentials enabled (boolean)
- `NOTES`: Notes (string)
- `VEHICLELICNUM`: Car license number (string)
- `VEHICLETAGNUM`: Car tag number (string)
- `CARDSTATUS`: Card status name.

- `UDF1...UDF20`: User defined fields (up to a maximum of 20).
- `HOTSTAMP` (optional): Returns the person record associated with this hot stamp number.
- `WANTCREDENTIALID`: Returns the credential ID for every credential associated with a returned person record. The credential ID is an alias for the actual credential number.

  As a security measure, credential IDs can be retrieved and stored in a client system in place of the encoded numbers and/or hotstamp numbers. This will allow people to manage credentials from the client system without seeing the actual credential numbers.

  USE **AddCredential** TO RETRIEVE THE CREDENTIAL ID FOR A CREDENTIAL WHEN IT IS ADDED.
- `ACCESSLEVEL` (optional): Return all person records associated with this access level.
- `OLDESTLASTMOD` (optional): Do not retrieve records whose last modification date is older than this time. If not specified, the oldest available records are returned. See *Date Formats*.
- `NEWESTLASTMOD` (optional): Do not retrieve records whose last modification date is more recent than this time. If not specified, the default is the current date and time. See *Date Formats*.
- `DELETED`: Specify `ALL`, `TRUE`, or `FALSE`:
  - `ALL`: Both *deleted* and *undeleted* person records are returned.
  - `TRUE`: Only *deleted* person records are returned.
  - `FALSE`: Only *undeleted* person records are returned.
- `ALLPARTITIONS`: If `TRUE`, the search incudes people in other partitions. If `FALSE` (or no parameter is specified), searches in the current partition only.

  The `ALLPARTITIONS` parameter requires *User Login Authentication*, and the account the API client uses to log in must have full system setup privileges.
- `CASEINSENSITIVE`: Specify "TRUE" or "FALSE" in the command (the default is "FALSE"):
  - TRUE: First and last name fields are retrieved using a case-insensitive search.
  - FALSE: No parameter is specified, first and last name fields are retrieved using a case-sensitive search.
- `WILDCARDSEARCH`: Specify "`TRUE`" or "`FALSE`" in the command (the default is "`FALSE`"):
  - `TRUE`: Person records corresponding to last names, first names, or User Defined Fields (UDFs) that contain the specified strings in the **SearchPersonData** call are returned.

- FALSE (or no parameter is specified): A non-wildcard search is performed.

  For example, if "TRUE" is specified for WILDCARDSEARCH, and the string entered for FIRSTNAME is "Fr," person records for people with the last names Frank, Frost, Friedman, and so forth are returned.

PARTITIONKEY (Global API Only): The partition to which the **SearchPersonData** call applies when the application has not been switched to a partition using the **SwitchPartition** command. This input is not accepted in the API. Based on the ALLPARTITIONS input parameter, all partitions or the current user's partition is searched upon.

- ACCESSLEVELDETAILS: A value of 0 indicates that only access level names are to be returned. A value of 1 indicates the returned access level list is to contain detailed information. Each <ACCESSLEVEL> block will contain:
  - ACCESSLEVELNAME: Name of the access level.
  - ACTDATE: Activation date for the access level.
  - EXPDATE: Expiration date for the access level.
  - AUTOREMOVE: A VALUE OF 1 MEANS THE ACCESS LEVEL WILL BE REMOVED WHEN ITS EXPIRATION DATE HAS BEEN REACHED. A VALUE OF 0 MEANS THE ACCESS LEVEL WILL NOT BE REMOVED WHEN ITS EXPIRATION DATE HAS BEEN REACHED.
- RAWCARDNUMBER: SPECIFY "TRUE" OR "FALSE" IN THE COMMAND:
  - TRUE: THE RAW CARD NUMBER FOR EACH ACCESS CARD IS RETURNED.
  - FALSE (OR NO PARAMETER IS SPECIFIED): NO RAW CARD NUMBERS ARE RETURNED.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Includes a DETAILS element block. The DETAILS element block includes the following element blocks:
  - PEOPLE: Element block containing the PERSON element block:
  - PERSON: Contains a PERSON element block for each person record retrieved based on the search criteria. The PERSON element block contains the following additional elements and element blocks:
  - PERSONID: Person ID.
  - FIRSTNAME: First name.
  - MIDDLENAME: Middle name.
  - LASTNAME: Last name.
  - ACTDATE: Activation date. See *Date Formats*.
  - EXPDATE: Expiration date. See *Date Formats*.
  - UFD1 through UFD20.
  - PIN: PIN number that may be required for a card reader with a numeric pad.

- NOTES: Notes (string).
- PARTITION: Partition of person record.
- DELETED: Specifies if the person record has been deleted (TRUE if deleted or FALSE).
- PICTUREURL: Name of file for picture data.
- BADGELAYOUT: Name of the photo ID badging layout file.
- LASTEDIT: Date and time the contents of the person record were last changed, in YYYY-MM-DD HH:MM:SS format.

  Note: Ignore the LASTMOD date and time, which will not necessarily be related to a content change.
- CONTACTPHONE: Office phone number.
- CONTACTEMAIL: Office email address.
- CONTACTSMSEMAIL: Office SMS email address.
- CONTACTLOCATION: Emergency contact location.
- OTHERCONTACTNAME: Emergency contact name.
- OTHERCONTACTPHONE1: Emergency contact phone number.
- OTHERCONTACTPHONE2: Alternate emergency contact phone number.
- VEHICLES: Element blocks containing a VEHICLE element block for each vehicle defined in the person record:
  - VEHICLECOLOR
  - VEHICLEMAKE
  - VEHICLEMODEL
  - VEHICLESTATE
  - VEHICLELICNUM
  - VEHICLETAGNUM
- ACCESSLEVELS: Contains an ACCESSLEVEL element for each ACCESS LEVEL defined in the person record.

  For each person, the list of access levels and access cards associated with that person are returned. If there are no access levels, none are returned.
- ACCESSCARDS: Contains an ACCESSCARD element block for each credential:
  - ENCODEDNUM
  - HOTSTAMP
  - CREDENTIALID: Returned if the WANTCREDENTIALID parameter is included in the call.
  - CARDFORMAT

- RAWCARDNUMBER: Returns the raw card number if TRUE is specified in the calling parameter.
- DISABLED: Returns 1 if the credential is currently marked disabled or 0 if it is not.
- CARDSTATUS: Status of the credential (refer to the UI for valid status values).
- CARDEXPDATE: Expiration date of credential. See *Date Formats*.

▪ NEXTKEY: This is returned with a value of -1, if there are no more people to return, or a specific value greater than zero that can be used in the next **SearchPersonData** call as the STARTFROMKEY value.

• FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:

Example

SearchPersonData call to retrieve information from a person record with a person ID value of 30001:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="SearchPersonData" num="1">
       <PARAMS>
          <PERSONID>30001</PERSONID>
          <WANTCREDENTIALID>1</WANTCREDENTIALID>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

In the example, the **SearchPersonData** call returns a CODE of SUCCESS. The DETAILS contain data from the person record with a person ID value of 30001.

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="SearchPersonData" num="1">
       <CODE>SUCCESS</CODE>
       <DETAILS>
          <PEOPLE>
          <PERSON>
             <PERSONID>30001</PERSONID>
             <FIRSTNAME>John</FIRSTNAME>
             <MIDDLENAME>V</MIDDLENAME>
             <LASTNAME>Smith</LASTNAME>
             <ACTDATE>2016-09-11 00:00:00</ACTDATE>
             <EXPDATE>2024-01-01 00:00:00</EXPDATE>
             <UDF1>Added User Defined Field 1</UDF1>
             <UDF2>Added User Defined Field 2</UDF2>
             <UDF3>Added User Defined Field 3</UDF3>
             <UDF4>Added User Defined Field 4</UDF4>
             <UDF5>Added User Defined Field 5</UDF5>
             <PIN>1111</PIN>
             <NOTES>John rides a motorcycle</NOTES>
             <PARTITION>Master</PARTITION>
             <DELETED>FALSE</DELETED>
             <PICTUREURL>john smith.jpg</PICTUREURL>
             <REGIONPRIVILEGE>1</REGIONPRIVILEGE>
             <BADGELAYOUT>smith.idpz</BADGELAYOUT
```

```
            <LASTMOD>2016-09-09 14:39:46.024999</LASTMOD>
            <CONTACTPHONE>999-123-4567</CONTACTPHONE>
            <CONTACTEMAIL>sford@yahoo.com</CONTACTEMAIL>
            <CONTACTSMSEMAIL>sford@s2sys.com</CONTACTSMSEMAIL>
            <CONTACTLOCATION>Building 1</CONTACTLOCATION>
            <OTHERCONTACTNAME>Mary Brown</OTHERCONTACTNAME>
            <OTHERCONTACTPHONE1>999-123-4568</OTHERCONTACTPHONE1>
            <OTHERCONTACTPHONE2>999-123-4569</OTHERCONTACTPHONE2>
            <VEHICLES>
               <VEHICLE>
                  <VEHICLECOLOR>Blue</VEHICLECOLOR>
                  <VEHICLEMAKE>Honda</VEHICLEMAKE>
                  <VEHICLEMODEL>Honda CRV</VEHICLEMODEL>
                  <VEHICLESTATE>MA</VEHICLESTATE>
                  <VEHICLELICNUM>123 PGA</VEHICLELICNUM>
                  <VEHICLETAGNUM>TAG 3000</VEHICLETAGNUM>
               </VEHICLE>
               <VEHICLE>
                  <VEHICLECOLOR>Red</VEHICLECOLOR>
                  <VEHICLEMAKE>Buggatti</VEHICLEMAKE>
                  <VEHICLEMODEL>XV32</VEHICLEMODEL>
                  <VEHICLESTATE>MA</VEHICLESTATE>
                  <VEHICLELICNUM>456 PGA</VEHICLELICNUM>
                  <VEHICLETAGNUM>TAG 3001</VEHICLETAGNUM>
               </VEHICLE>
            </VEHICLES>
            <ACCESSLEVELS>
               <ACCESSLEVEL>Access level 1</ACCESSLEVEL>
               <ACCESSLEVEL>Access level 2</ACCESSLEVEL>
            </ACCESSLEVELS>
            <ACCESSCARDS>
               <ACCESSCARD>
                  <CREDENTIALID>3</CREDENTIALID>
                  <CARDFORMAT>26 bit Wiegand</CARDFORMAT>
                  <DISABLED>0</DISABLED>
                  <CARDSTATUS>Active</CARDSTATUS>
                  <CARDEXPDATE>
               </ACCESSCARD>
            </ACCESSCARDS>
         </PERSON>
         </PEOPLE>
         <NEXTKEY>-1</NEXTKEY>
      </DETAILS>
   </RESPONSE>
</NETBOX>
```

The picture data file (returned as text between the `PICTUREURL` elements) is stored in the directory, `/usr/local/s2/web/upload/pics`, on the system. The returned `REGIONPRIVILEGE` refers to the regional anti-passback privilege.

SearchPersonData call with the search criteria of `LASTNAME` with a value of Smith:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="SearchPersonData" num="1">
        <PARAMS>
           <LASTNAME>Smith</LASTNAME>
        </PARAMS>
     </COMMAND>
</NETBOX-API
```

Successful response to the **SearchPersonData** call returns all occurrences of person records with the last name of Smith:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="SearchPersonData" num="1">
        <CODE>SUCCESS</CODE>
        <DETAILS>
            <PEOPLE>
                <PERSON>
                    <PERSONID>1007</PERSONID>
                    <FIRSTNAME>Frank</FIRSTNAME>
                    <MIDDLENAME></MIDDLENAME>
                    <LASTNAME>Smith</LASTNAME>
                          .
                          .
                          .
                 </PERSON>
                <PERSON>
                    <PERSONID>1011</PERSONID>
                    <FIRSTNAME>Roger</FIRSTNAME>
                    <MIDDLENAME></MIDDLENAME>
                    <LASTNAME>Smith</LASTNAME>
                          .
                          .
                          .
                 </PERSON>
                <PERSON>
                    <PERSONID>1012</PERSONID>
                    <FIRSTNAME>Betty</FIRSTNAME>
                    <MIDDLENAME></MIDDLENAME>
                    <LASTNAME>Smith</LASTNAME>
                          .
                          .
                          .
                </PERSON>
            </PEOPLE>
            <NEXTKEY>-1</NEXTKEY>
        </DETAILS>
    </RESPONSE>
</NETBOX>
```

SetThreatLevel

The **SetThreatLevel** command sets the threat level.

Calling Parameters

- `LEVELNAME`: Name of threat level.

Response

This command returns `SUCCESS` or `FAIL` in the `CODE` element:

- `SUCCESS`: Returns no additional elements.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Not Found
  - Missing threat level

Example

SetThreatLevel call to request that the threat level is set to "Intruder Alert":

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="SetThreatLevel" num="1">
       <PARAMS>
          <LEVELNAME>Intruder Alert</LEVELNAME>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **SetThreatLevel** call indicates that the threat level was set:

```
<NETBOX> sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="SetThreatLevel" num="1">
       <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

StreamEvents

The **StreamEvents** command retrieves event notifications.

The **StreamEvents** command is not covered in the API.

StreamEvents command is accepted at address:

```
http://<NetBox IP Address>//nbws/goforms/nbapi
```

StreamEvents establishes a connection to the system that remains open until either the program or the system closes it. As events occur, the system sends data to inform the application of events (activity).

See *The Event API* for a detailed discussion of how to stream and filter events, including element descriptions.

Calling Parameters

- `TAGNAMES`: (optional) element containing a set of tags elements (e.g. `ACNAME`, `ACTIVITYID`, …) with optional `FILTERS`. A `FILTERS` element on a tag contains one or more `FILTER` elements that define a value to filter on. If multiple filters are specified for an element, then the comparison operation is a logical OR operation among all of the filters.

Example

The following example contains a `TAGNAMES` element block, which identifies a list of fields that may be included in the event responses. The partition name, `PARTNAME` element includes a `FILTER` element, which filters on the partition name of "Master."

```
<NETBOX-API>
    <COMMAND name="StreamEvents">
       <PARAMS>
          <TAGNAMES>
             <ACNAME />
             <ACTIVITYID />
             <CDT />
             <DESCNAME />
             <LOGINADDRESS />
             <PERSONNAME />
             <DETAIL />
             <PARTNAME>
                <FILTERS>
                   <FILTER>Master</FILTER>
                </FILTERS>
             </PARTNAME>
          </TAGNAMES>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Upon successful invocation, responses will stream to the application, such as:

```
<NETBOX>
     <RESPONSE command="StreamEvents">
        <EVENT>
           <ACTIVITYID>976</ACTIVITYID>
           <DESCNAME><![CDATA[Event activated]]></DESCNAME>
           <CDT>2016-11-03 17:00:53.942949</CDT>
           <PARTNAME><![CDATA[Master]]></PARTNAME>
        </EVENT>
     </RESPONSE>
</NETBOX>

<NETBOX>
     <RESPONSE command="StreamEvents">
        <EVENT>
           <ACTIVITYID>977</ACTIVITYID>
           <DESCNAME><![CDATA[Event normal]]></DESCNAME>
           <CDT>2016-11-03 17:01:02.694311</CDT>
           <PARTNAME><![CDATA[Master]]></PARTNAME>
        </EVENT>
     </RESPONSE>
</NETBOX>
```

Multiple events can be included in a response. The elements included in the `EVENT` element are dependent upon the specific event type. Not all elements are included in every notification.

Event notifications commence starting from the issuance of the request. Past events are not delivered through this interface. The API program is responsible for re-issuing the request in the event that it gets disconnected from the system.

Every ten seconds, a keep-alive response containing zero events is streamed to the client via the **StreamEvents** command.

SwitchPartition

The **SwitchPartition** command switches to a specified partition. Subsequent API commands make modifications to that partition. The **SwitchPartition** does not change the partition currently displayed in the user interface.

The **SwitchPartition** command requires *User Login Authentication* and the account that the API client uses to log in must have full system setup, partition setup or user role mapping privilege for the specific partition.

Calling Parameters

- PARTITIONKEY: The partition key.

  A **SwitchPartition** call requires a PARTITIONKEY parameter. The PARTITIONKEY value corresponds to a partition defined in the system database. After a **SwitchPartition** call is issued, all commands apply to that partition.

  For Global API only:

  - An additional PARTITIONKEY value of 0, which specifies that the current login session, initiated by the most recent **Login** call, has no partition assigned. 0 is the default value when a **Login** call is issued.
  - After a **Login** call is issued, the partition remains unassigned (PARTITIONKEY = 0) until a **SwitchPartition** call is issued with a new PARTITIONKEY value.

    Use **GetPartitions** to retrieve the list of PARTITIONKEY values.

Response

This command returns SUCCESS, FAIL, or NOT FOUND in the CODE element:

- SUCCESS: Returns no additional elements.
- FAIL: Returns an error message in the ERRMSG element in the DETAILS element block:
  - Invalid partition key
  - Missing partition key

Example

SwitchPartition call to switch to the partition named "Third Partition" associated with a PARTITIONKEY with the value of 3:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="SwitchPartition" num="1">
       <PARAMS>
          <PARTITIONKEY>3</PARTITIONKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **SwitchPartition** call indicates that the current partition was changed:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
     <RESPONSE command="SwitchPartition" num="1">
        <CODE>SUCCESS</CODE>
     </RESPONSE>
</NETBOX>
```

All subsequent commands will apply to the changed partition.

TriggerEvent

The **TriggerEvent** command triggers an event to activate or removes the trigger to affect deactivation.

The **TriggerEvent** command is not covered as part of NBAPI version 2.

The **TriggerEvent** command is accepted at address:

```
http://<NetBox IP Address>/nbws/goforms/nbapi
```

Use **ListEvents** to retrieve the list of events to trigger.

See *The Event API* for a detailed discussion of trigger events.

TriggerEvent requires a sessionid.

Calling Parameters

- `EVENTNAME`: The name of the event to activate or deactivate.
- `EVENTACTION`: Either `ACTIVATE` or `DEACTIVATE`.
- `PARTITIONID`: The partition of the event to activate or deactivate. If `PARTITIONID` is not specified, the Master partition is assumed.

Response

This command returns `SUCCESS`, `FAIL` in the `CODE` element:

- `SUCCESS`: Returns no additional elements.
- `FAIL`: Returns an error message in the `ERRMSG` element in the `DETAILS` element block:
  - Event name does not exist.
  - Partition does not exist.
  - Invalid event action – Must be *either ACTIVATE or DEACTIVATE.*
  - Event already active.
  - Event not active.
  - Activate failed.
  - Error during execution of a trigger event.

Example

TriggerEvent call to activate the "158_backDoorForcedEvent" event.

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
     <COMMAND name="TriggerEvent">
```

```
    <PARAMS>
       <EVENTNAME>158_backDoorForcedEvent</EVENTNAME>
       <EVENTACTION>ACTIVATE</EVENTACTION>
       <PARTITIONID>1</PARTITIONID>
    </PARAMS>
    </COMMAND>
</NETBOX-API>
```

The successful response to the TriggerEvent command:

```
<NETBOX>
    <RESPONSE command="TriggerEvent">
        <CODE>SUCCESS</CODE>
    </RESPONSE>
</NETBOX>
```

UnlockPortal

The **UnlockPortal** command requests that a portal specified by a PORTALKEY be set to the Extended Unlock state.

Use **GetPortalGroups** to retrieve the list of PORTALKEY values.

Use **LockPortal** to request that a portal be set to the Ready (locked) state.

Use **MomentaryUnlockPortal** to request that a portal be momentarily unlocked.

Calling Parameters

- PORTALKEY: Key associated with the portal to be unlocked.

Response

This command returns SUCCESS or FAIL in the CODE element:

- SUCCESS: Returns the PORTALKEY associated with the portal to be unlocked in the DETAILS element block.
- FAIL: Returns the following error message in the ERRMSG element in the DETAILS element block:
  - Invalid portal key – the key provided does not match a portal.
  - Missing PORTALKEY
  - Portal not online or unreachable – the resource associated with the portal is unavailable.
  - Portal state not changed
  - Command not supported for this device type
  - SQL function returned unexpected value

Example

UnlockPortal call to request that the portal associated with a PORTALKEY with a value of 7 is unlocked:

```
<NETBOX-API sessionid="Qq7XhuKZuTdYPNcCw">
    <COMMAND name="UnlockPortal" num="1">
       <PARAMS>
          <PORTALKEY>7</PORTALKEY>
       </PARAMS>
    </COMMAND>
</NETBOX-API>
```

Successful response to the **UnlockPortal** call indicates that the portal has been unlocked:

```
<NETBOX sessionid="Qq7XhuKZuTdYPNcCw">
    <RESPONSE command="UnlockPortal" num="1">
       <CODE>SUCCESS</CODE>
          <DETAILS>
             <PORTALKEY>7</PORTALKEY>
          </DETAILS>
```

```
    </RESPONSE>
</NETBOX>
```

Deprecated Commands

This section contains a list of commands that are no longer supported.

If your program code contains these commands, you need to replace them with the preferred commands.

Deprecated Command	Replace With
EditPerson	**AddPerson and ModifyPerson**
EditThreatLevel	**AddThreatLevel and ModifyThreatLevel**
EditThreatLevelGroup	**AddThreatLevelGroup and ModifyThreatLevelGroup**
GetAccessDataLog	**GetAccessHistory**
GetAccessCardDetails	**GetCardAccessDetails**
LoginUserName	**N/A**
LoginUserPassword	**N/A**